

Aprenda rápidamente a programar

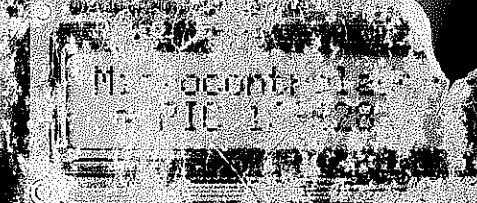
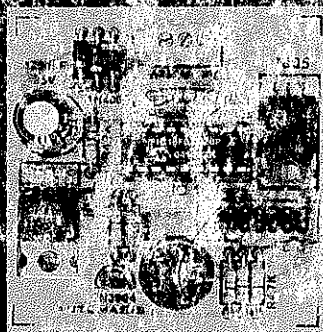
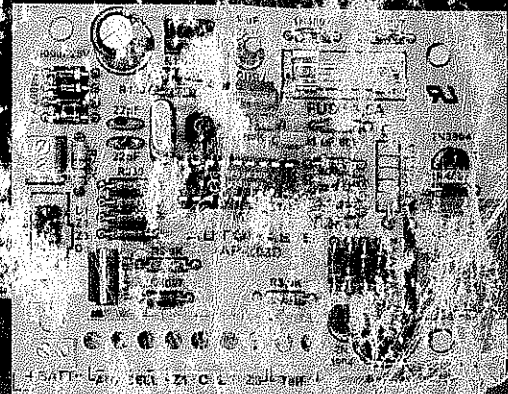
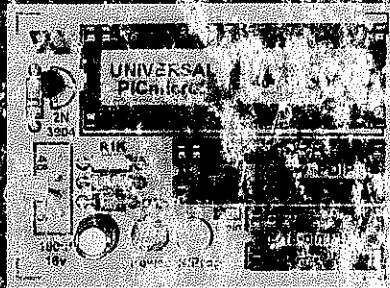
Microcontroladores

PIC

16F62X
16F81X
12F6XX

En lenguaje BASIC

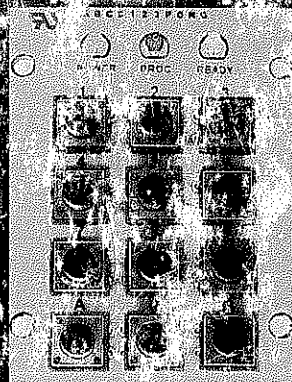
CARLOS A. REYES



CONSTRUYA SU PROPIO
GRABADOR DE PICHOS

- ALARMA PARA EL HOGAR
- CONTROL DE ACCESOS
- MANEJO DE MÓDULOS LCD
- MANEJO DE DISPLAY DE 7 SEG.
- CONTROL DE MOTORES DC PASO A PASO
- TACÓMETRO DIGITAL
- COMUNICACIÓN SERIAL
- GENERACIÓN DE SONIDO
- GENERACIÓN DE TONOS DTMF, ETC.

FABRICACIÓN DE CIRCUITO IMPRESO CON
TRANSFERENCIA TÉRMICA



MICROCHIP

INCLUYE
CD ROM

INCLUYE LÁMINA PARA
EL GRABADOR DE PICHOS

CARLOS A. REYES

aprenda a programar microcontroladores PIC

\$29.85
\$457a

PRÓLOGO

En la década de los años 80 mientras se estrenaba una serie de películas de ciencia-ficción, hubo una que despertó mucha fascinación entre los estudiantes y aficionados a la electrónica, justamente la película TERMINATOR, que con un impresionante despliegue de tecnología este robot del futuro nos dejó cautivados a muchos, precisamente en una de las escenas es donde hacen la presentación de un diminuto microchip, nos estaban mostrando el futuro de la tecnología del mañana. Hoy a principios del siglo XXI vemos que esa realidad no está muy distante ya que solo nuestra imaginación es el límite para la creación. Si analizamos nuestro alrededor veremos que ya están presentes en muchos de los equipos electrónicos, hoy podemos encontrar en un hogar hasta 37 microcontroladores, y cada día seguirán implementándose en nuevos equipos.

El microcontrolador es hoy en día la piedra angular de lo que será el imperio tecnológico del siglo XXI, dentro de unos años estaremos rodeados de microcontroladores y serán indispensables para la vida cotidiana. Si analizamos actualmente lo que pasaría si dejaran de funcionar por un minuto, nos daremos cuenta que no habría comunicación, las centrales telefónicas, celulares y radios dejarían de funcionar, lo mismo sucedería con; los computadores, los satélites y con ello el comercio marítimo entraría en caos, el tráfico aéreo estaría en peligro, los mismos aviones no podrían volar sin sus instrumentos de navegación, los sistemas electrónicos de los vehículos fallarían, las fabricas paralizarían su producción, en los hospitales muchos equipos electrónicos fallarían, las centrales eléctricas dejarían de suministrar energía, en el hogar dejarían de funcionar los sistemas de seguridad, vigilancia y de incendio, los ascensores, el microondas, la lavadora, el televisor, la radio el DVD, etc. en definitiva sería como que si el mundo se detendría aunque seguiría girando por supuesto.

Por eso es tan importante el conocimiento y la utilización de estos circuitos integrados que en un principio lo llamaron microcomputadores para en la actualidad conocerlos como los poderosos microcontroladores. El que en este libro trataremos es específicamente el microcontrolador PIC de Microchip Technology Inc. ya que es el fabricante que lleva el liderazgo por su bajo costo, fácil programación y la gran disponibilidad de modelos a elegir según sea las necesidades.

La metodología de enseñanza será hacer el proyecto y poner en funcionamiento, para luego dar sus debidas explicaciones bien detalladas ya que también está orientada a principiantes, podría parecerles muy obvio pero para alguien esa explicación no estaría demás, yo mismo he tenido problemas con algunos libros al no poder pasar más allá del primer ejercicio, aunque el libro decía que era para principiantes, y sin contar que me cansaba leyendo tanta teoría. Por eso no pretendo ponerles una *montaña de teoría* y causar desinterés en el lector, porque para ello existen muchos libros donde tratan más a fondo la estructura del microcontrolador, más bien intento entusiasmarles con proyectos de fácil aprendizaje y muy útil aplicación en su hogar, empresa y por qué no para realizar proyectos importantes de automatización de fábricas, ya que

se incluirá circuitos de control y manejo de Control computarizado con Visual Basic, al final de los capítulos se dará referencia de paginas web en donde se puede encontrar más información al respecto.

Ya que este libro fue realizado con el apoyo de prácticas reales, encontrarán la información y los consejos más importantes que puedo darles en base a las experiencias propias que he adquirido a la hora de montar mis propios proyectos, años y años de trabajo encontrarán en este resumido libro. Justamente este es el motivo más grande que me impulsó a escribir este libro. Un día mientras realizaba una práctica con los PIC'S, vino mi hijo de 5 años con sus preguntas de ¿Qué es esto?, ¿Para qué sirve?, ¿Cómo funciona?, etc. En ese momento me puse a pensar como podría enseñarle todo lo que sé y con su temprana edad como podría entenderlo, he sido profesor de microcontroladores por un tiempo y mi problema de audición que cada día se agrava me impide continuar con esta noble labor. Una vez un profesor de la primaria dijo; El objetivo del hombre en la vida era, sembrar un árbol tener un hijo y escribir un libro, el árbol significaba el trabajo y la contribución para el futuro del planeta, el hijo significaba la experiencia de ser padres y la continuación de la especie humana, el libro es nuestra experiencia y lo que nosotros aprendemos a lo largo de nuestro ciclo de vida, y lo resumimos para nuestros hijos, para que así puedan continuar con nuestro trabajo y avancen mucho más que nosotros. También una vez mi padre dijo; Cuando se muere ni siquiera la ropa que estamos puestos nos podemos llevar. Con estas palabras deduje si algún día llego a morir, todos mis conocimientos me los llevo, es lo único que se va conmigo así que eso no se puede dejar de herencia, por todas estas razones he pensado que la mejor obra y el mejor regalo que puedo dejar son las letras, es decir un libro, donde todos podrán aprender de mi, ya que en algún lugar siempre habrá una persona que necesita de mis conocimientos.

Adicionalmente para ayudar al lector se incluye en este libro una lámina para transferencia térmica del grabador de PIC'S y un CD con todos los ejercicios, diagramas, hojas de dato y fotografías a color de las placas que a lo largo del Capítulo 5 y 6 se van presentando, aquí podrán observar detenidamente y con total claridad cada una de las placas electrónicas (PCB), así como también podrán observar una secuencia fotográfica de cómo hacer placas de circuito impreso con el método de transferencia térmica. *Para utilizar el CD, únicamente copie todo el contenido del CD en una sola carpeta de su disco duro.*

Finalmente quiero expresar mi más sincero agradecimiento a Microchip por su valiosa ayuda tanto en información como en softwares de libre distribución, a mecanique por su valioso editor de textos que se distribuye gratuitamente por internet, a Bonny Gijzen por su programa IC-Prog, a microEngineering Labs. y a todos mis profesores y amigos que me han ayudado a despejar mis dudas, a mi esposa que tanto me apoyó en la realización de este libro, y de manera especial al Sr. Ing. Juan Bernardo Tamariz y la señorita Julia León de **Corpoimpex** distribuidores exclusivos de **Microchip** para Ecuador, por facilitarme las hojas de datos que se incluye en el CD, y permitirme utilizar el logo de **Microchip** en este libro, además de softwares, materiales, información y facilidades que me han brindado.

Quito, Ecuador,
Octubre del 2004

INTRODUCCIÓN

El Capítulo 1 trata de la descarga e instalación de los softwares necesarios para la programación de los PIC, mediante ilustraciones gráficas seguiremos paso a paso la descarga de cada uno de los programas, todo esto para que el lector no pueda perderse, recuerden ninguna explicación está demás, siempre habrá alguien que lo necesite.

En el capítulo 2 se da una breve teoría del microcontrolador, específicamente del PIC16F62X, y sus características más sobresalientes, además se da consejos muy útiles para el correcto funcionamiento del PIC.

En el capítulo 3 se enseña a configurar y programar en microcode, así como también a reconocer los tipos de errores que se pueden dar en la compilación del programa escrito.

En el Capítulo 4 se explica porqué escoger el compilador PicBasic Pro 2.44 y su comparación con el lenguaje acostumbrado el ensamblador, también formas diferentes de escribir los programas, y como grabar en el programa IC-prog, además se enseña a reconocer errores en la grabación del PIC.

En el Capítulo 5 iniciamos el aprendizaje con ejercicios básicos pero muy necesarios para poder comprender el funcionamiento del microcontrolador PIC. También cabe recalcar que es indispensable disponer de un grabador de PIC'S y así poder realizar cada uno de los 52 proyectos que se encuentran en este capítulo, como también se debe seguir en orden cada uno de los proyectos para poder entender más adelante, ya que cada vez serán más grandes pero no muy difíciles de entender.

En el Capítulo 6 se enseña a fabricar circuitos impresos, mediante un método sencillo y revolucionario: *la transferencia térmica*, aquí se indicará todos los pasos necesarios para la fabricación del grabador de PIC'S UNIVERSAL PICmicro3, para el cual se entrega conjuntamente con este libro una lámina para la transferencia térmica de las pistas y el screen de los elementos, además se enseñará técnicas de soldaduras y terminaremos con la fabricación de un chasis para sus proyectos, este le dará una buena presentación y acabado.

CAPÍTULO 1

DESCARGA E INSTALACIONES NECESARIAS PARA EL PIC

1. DESCARGA E INSTALACIÓN DE LOS SOFTWARES.

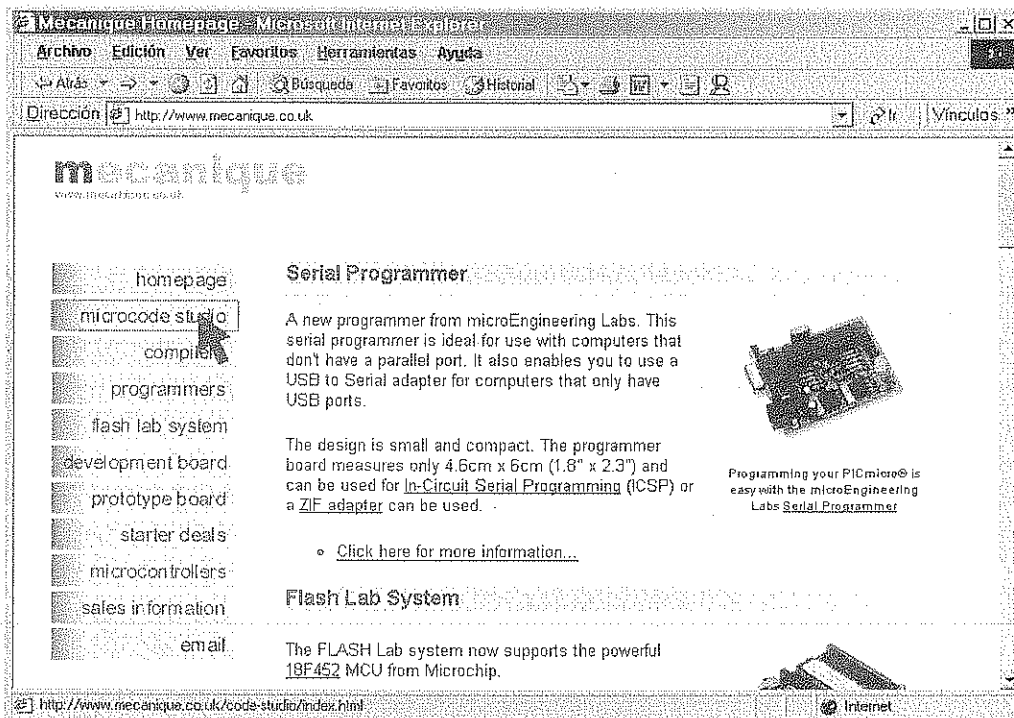
En este Capítulo se aprenderá a descargar los softwares necesarios para poder editar y compilar nuestros ejercicios, seguiremos paso a paso como descargar del internet.

Recuerden hay actualizaciones cada 6 meses por lo que debemos estar visitando estas páginas constantemente.

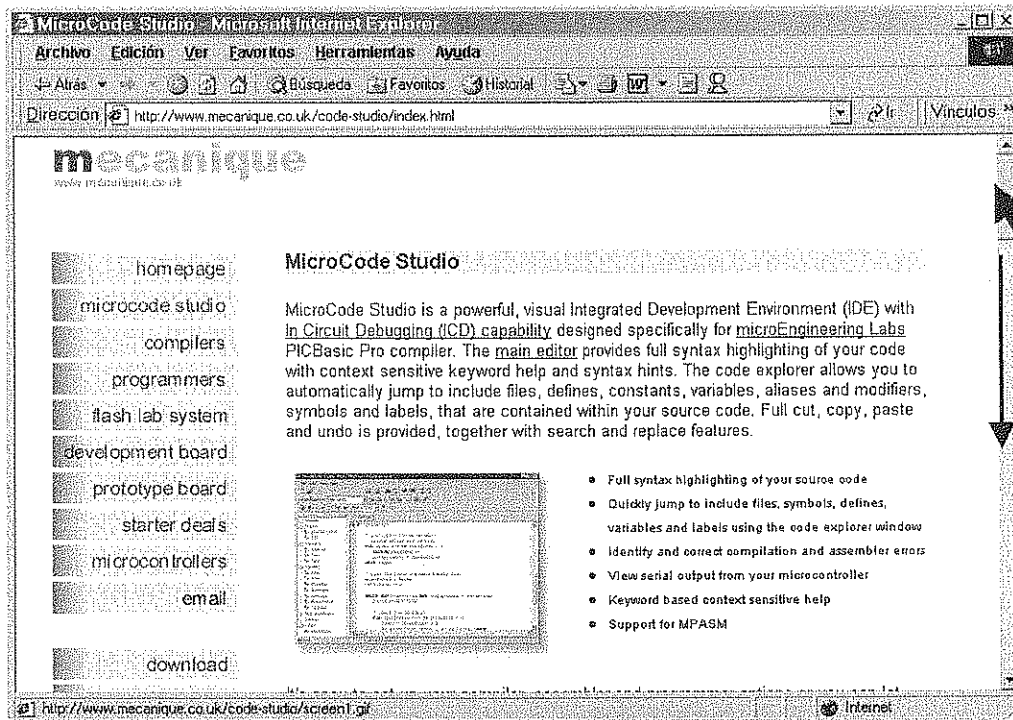
1.1 DESCARGA DEL PROGRAMA GRATUITO MICROCODE.

Este es el primer programa que vamos a descargar desde el internet, primero debemos ingresar en la página www.mecanique.co.uk y realizar los siguientes pasos:

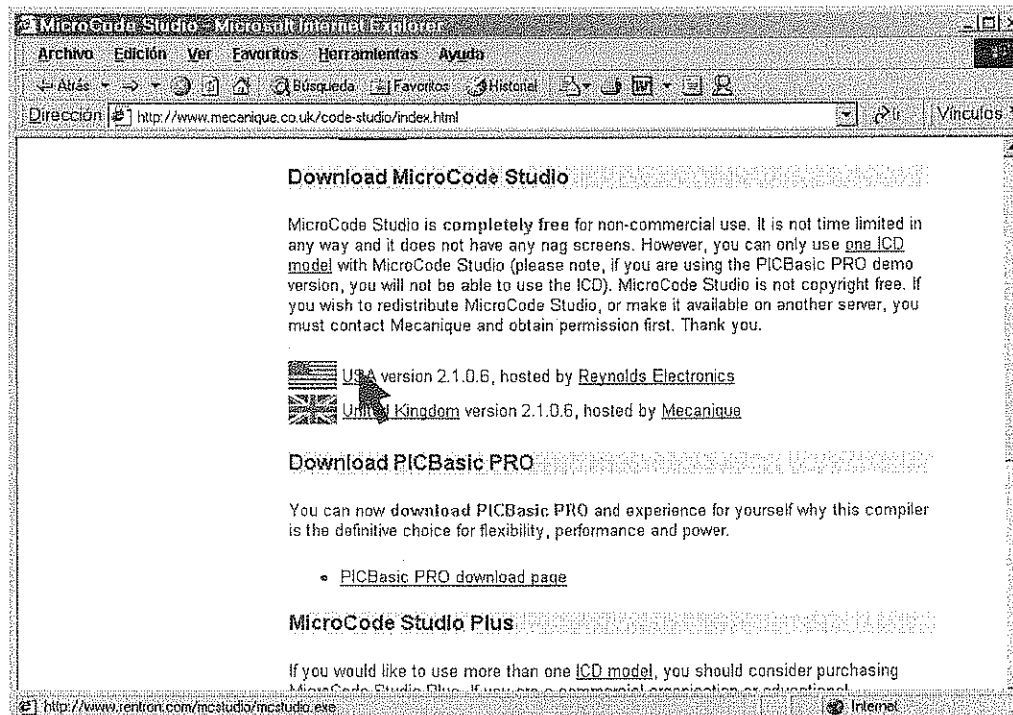
1a) Una vez abierto la página web nos deslizamos hasta donde dice microcode studio y damos un clic, como ilustra la siguiente figura:



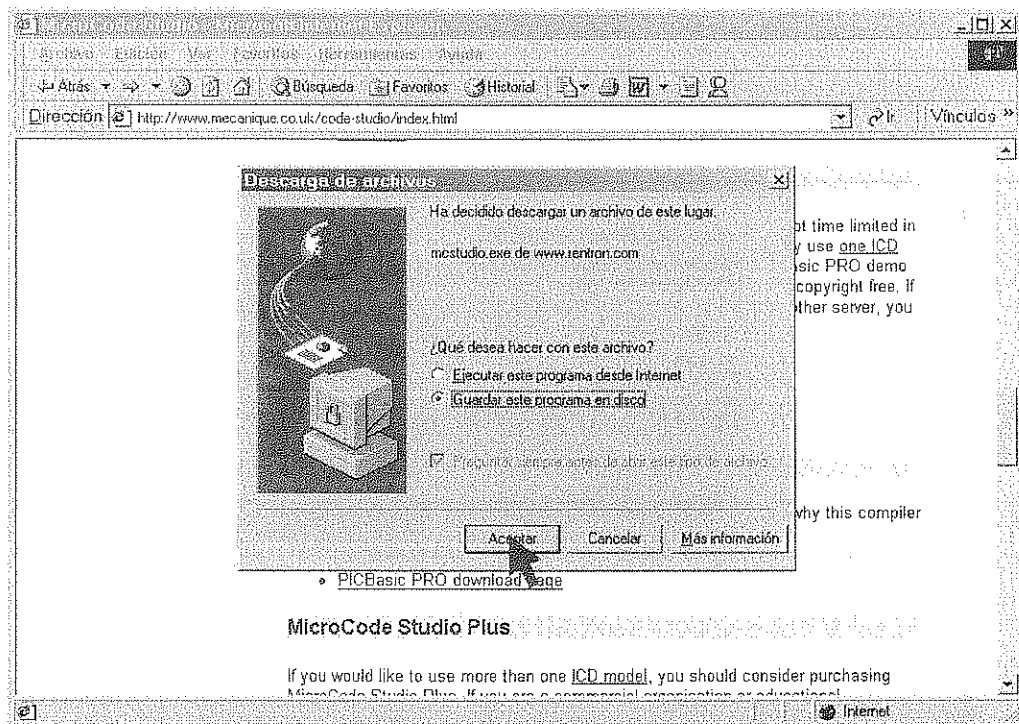
1b) Nos presentará esta nueva pantalla, deslizamos la barra de desplazamiento que está a la derecha de la pantalla hasta encontrar lo que viene a continuación en el siguiente literal:



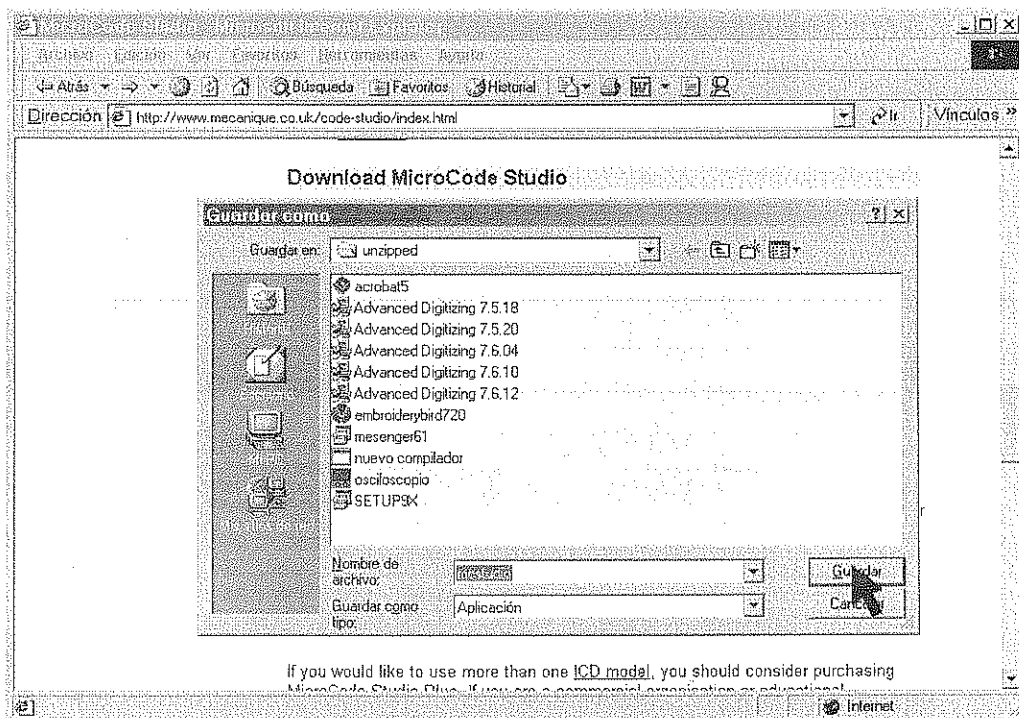
1c) En esta pantalla damos un clic en donde dice USA versión 2.1.0.6, y esperamos un momento para pasar al siguiente literal.



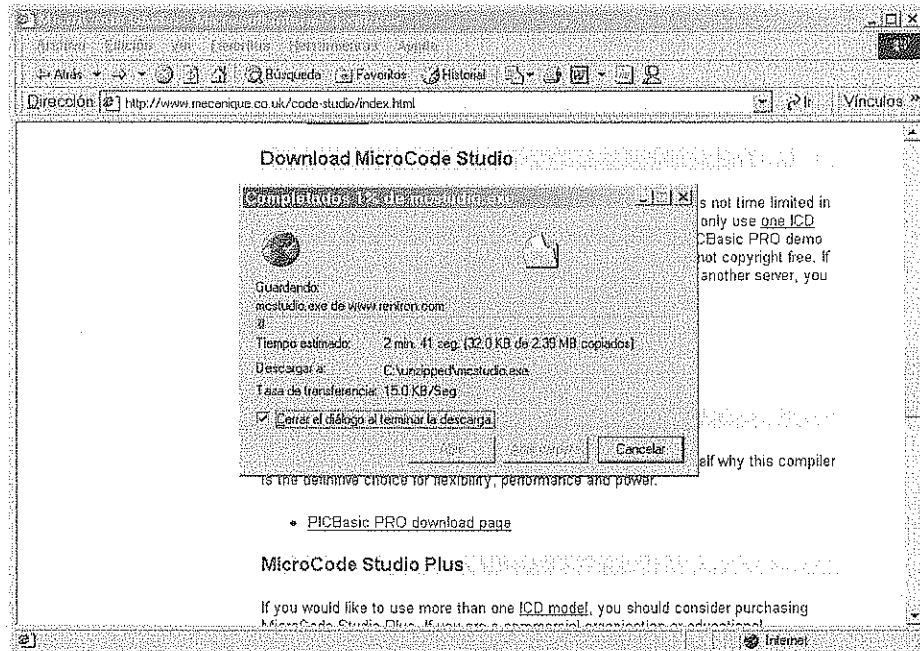
1d) En unos segundos saldrá una pantalla similar al siguiente gráfico, en donde sale seleccionado **Guardar este programa en disco**, solo damos un clic en aceptar.



1e) En instantes sale una pantalla similar al siguiente gráfico, en donde nos indica en que carpeta se va a descargar, si queremos le ubicamos en C:\unzipped\ o en cualquier carpeta que deseemos, también podemos cambiar el nombre del archivo, y luego de esto damos un clic en guardar.



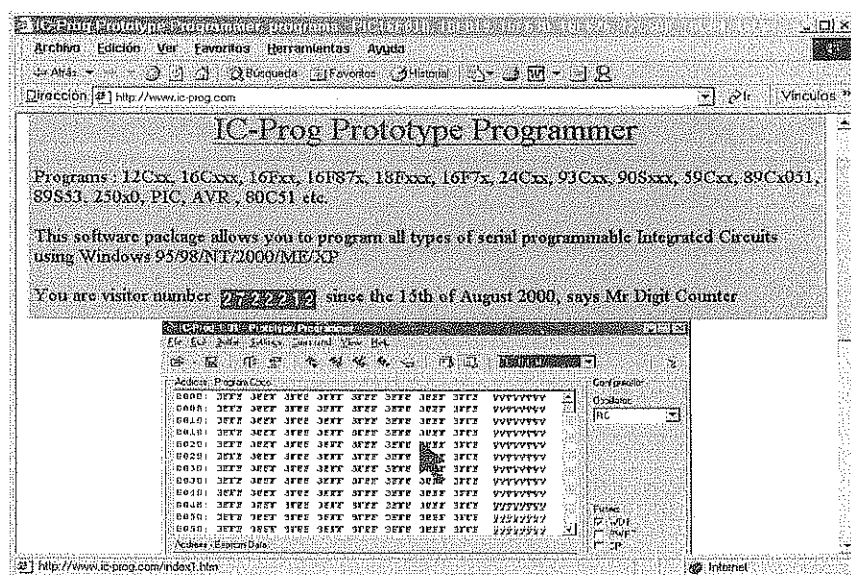
1f) Enseguida veremos una pantalla de descarga, aquí podemos marcar el recuadro de **Cerrar el diálogo al terminar la descarga** para que se cierre automáticamente al finalizar la descarga y nos emita un sonido, esta descarga puede tardar de 3 a 8 minutos, mientras hace esto podemos seguir descargando el siguiente programa.



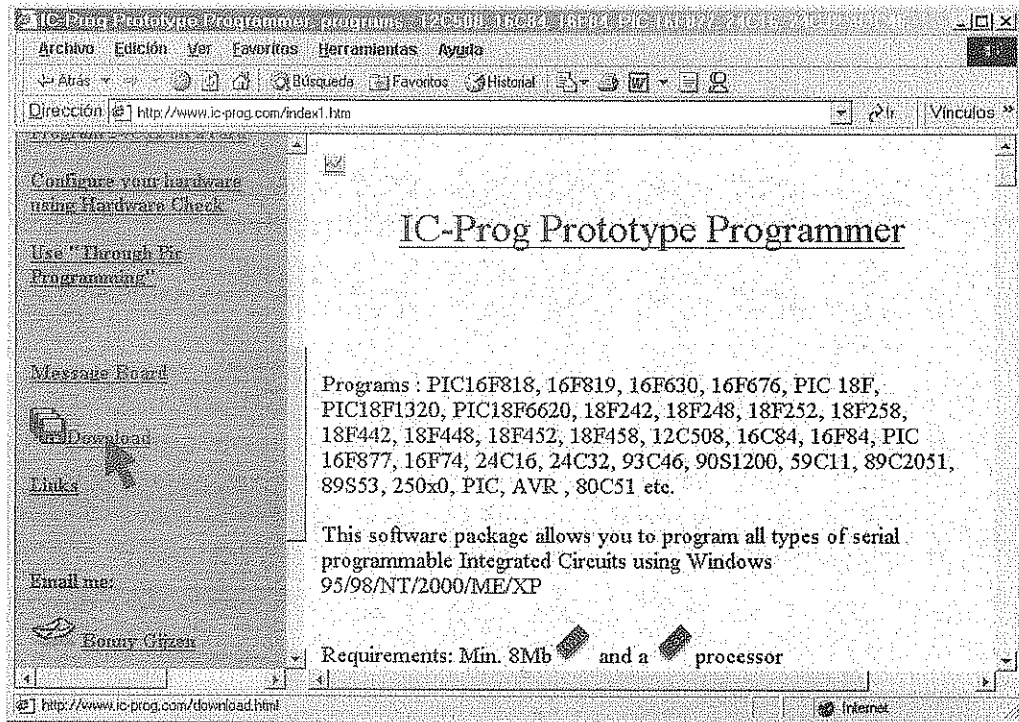
1.2 DESCARGA DEL PROGRAMADOR IC-prog Y EL DRIVE NT/2000/XP.

2a) Ingresamos a la página WWW.IC-prog.com, no olvidar la raya porque también hay otra página WWW.Icprog.com que no nos sirve, damos un clic en el medio del gráfico y esperamos.

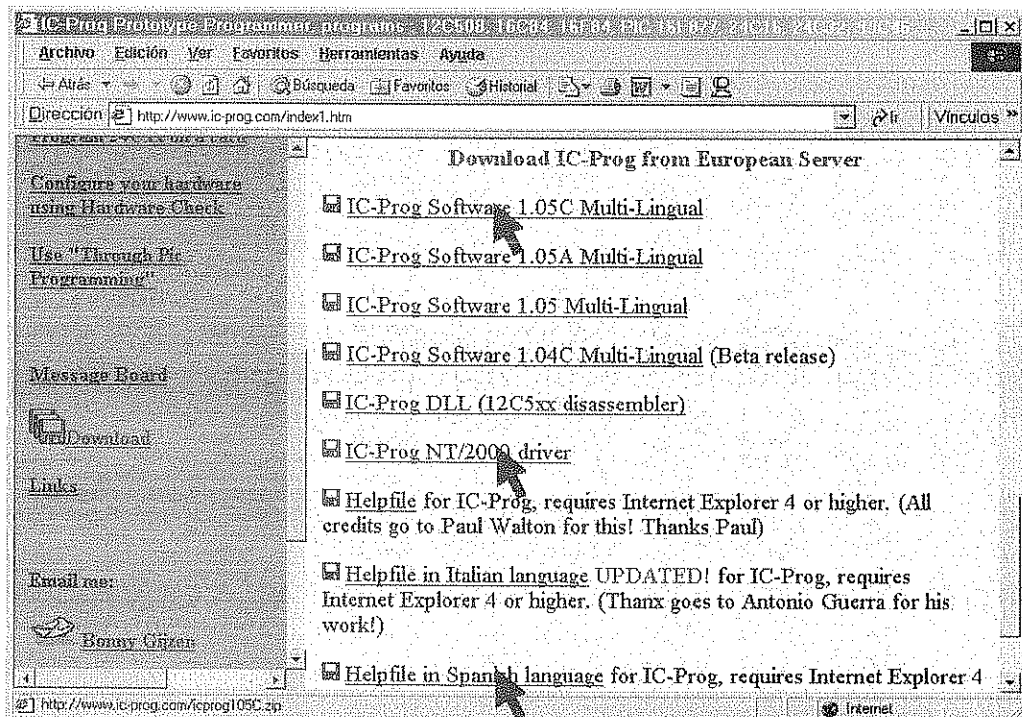
NOTA: Se puede descargar la versión 105c-a (PIC16F628A), en WWW.talkingelectronics.com en la sección Download (Files), icprog105c-a.zip.



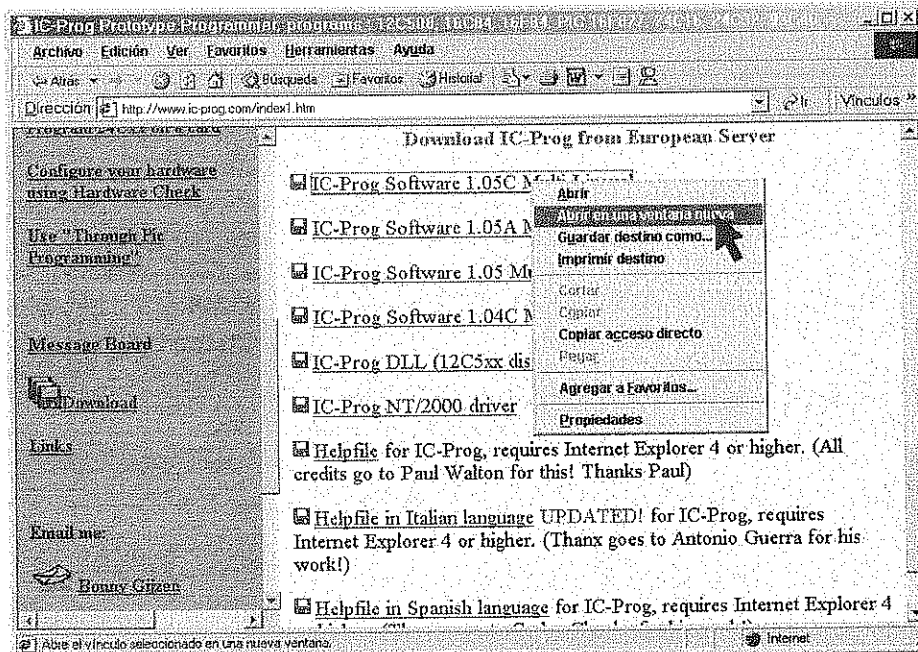
2b) Nos mostraran una pantalla similar al siguiente gráfico y damos clic en donde dice Download.



2c) En esta pantalla podemos descargar 3 archivos simultáneamente, el primero es el programa IC-Prog 1.05C, el segundo es el driver para windows NT/2000yXP que podríamos necesitar si tenemos estos sistemas operativos, y el tercero es una ayuda en español **Helpfile in Spanish Language**. Primero descargamos IC-Prog 1.05C como indica el siguiente literal:

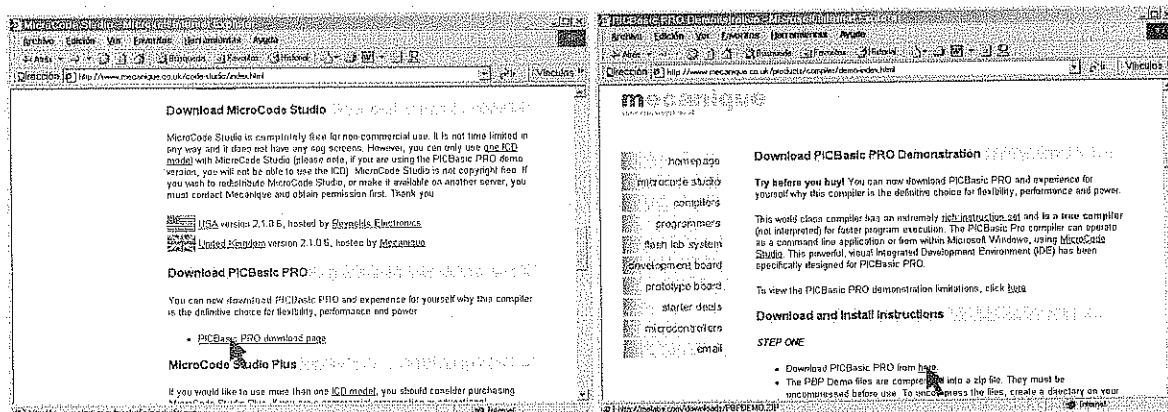


2d) Para poder descargar los 3 archivos simultáneamente, damos un clic en donde dice **IC-Prog 1.05C** con el botón *derecho* del mouse y luego un clic en donde dice **Abrir en una ventana nueva**, nos presentaran una ventana similar al del literal 1d) procedemos igualmente con el literal 1e) y cuando este en el literal 1f) lo minimizamos, y volveremos a la pantalla 2c) procedemos a descargar el driver NT/2000 y luego con la ayuda Helpfile Spanish Language.

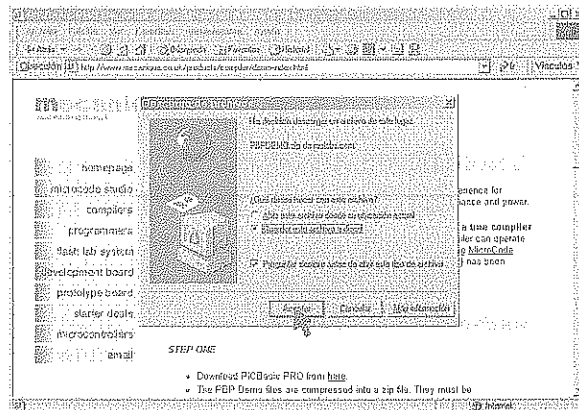


1.3 DESCARGA DE pbp 2.44 (PicBasic Pro ver 2.44)

Este es nuestro compilador que nos facilitará las cosas ya que disponemos un programador de PIC el IC-Prog 105C, también tenemos un editor de texto el microcode 2.1.0.6 y solo nos falta un programa compilador que se encargará de generar los archivos .HEX, necesario para poder grabar en un microcontrolador PIC. Para esto necesitamos adquirir el programa pbp 2.44 que cuesta alrededor de USD. 250, también nos facilitan una versión demo que lo podemos encontrar en la página www.mecanique.co.uk, en el literal 1c) observen que en la parte inferior dice **PicBasic PRO download page**, damos un clic y luego que se presente la nueva pantalla damos un clic donde dice here. **NOTA:** también se puede descargar una versión completa del compilador pbp 2.44 en la página WWW.sonsivri.com.

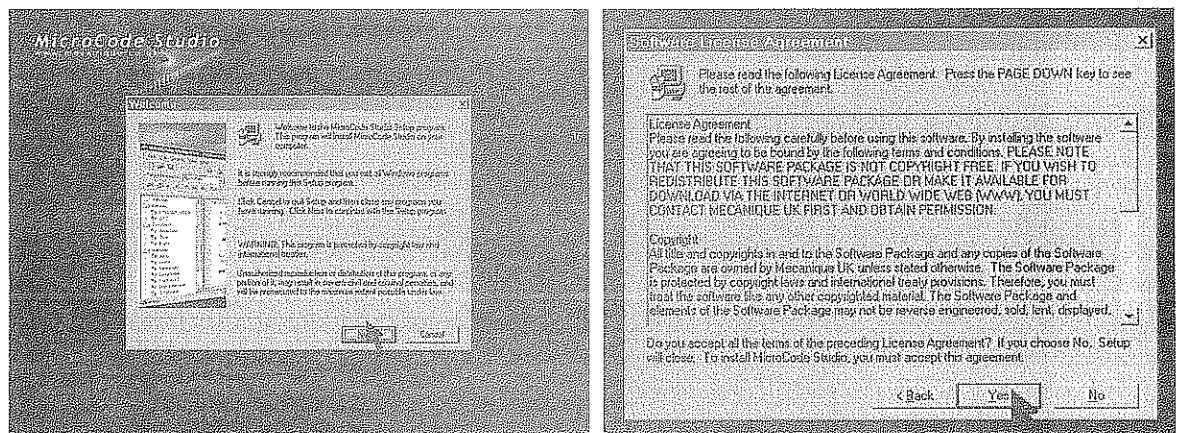


Esperamos a que salgan la pantalla de confirmación de descarga presionamos Aceptar, y luego procedemos como se aprendió en los casos anteriores.

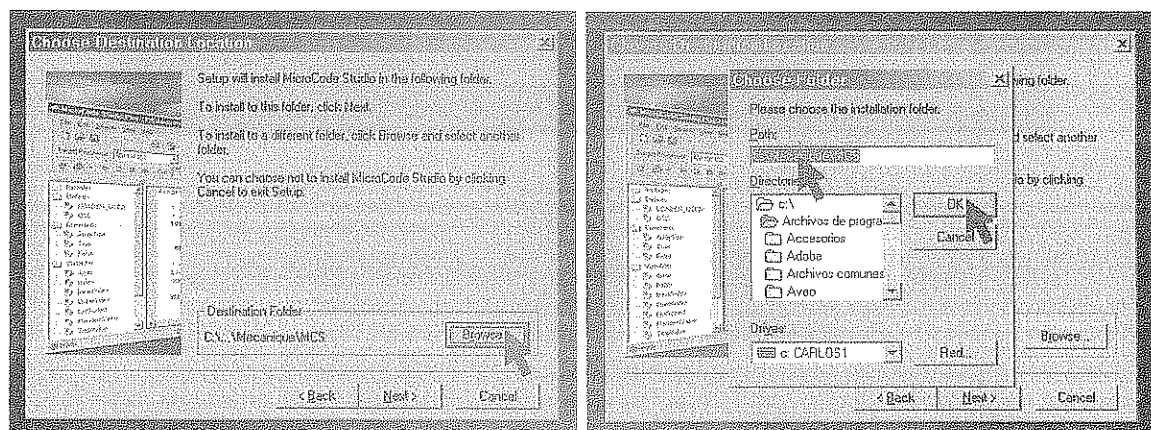


1.4 INSTALACIÓN DEL SOFTWARE MicroCode Studio.

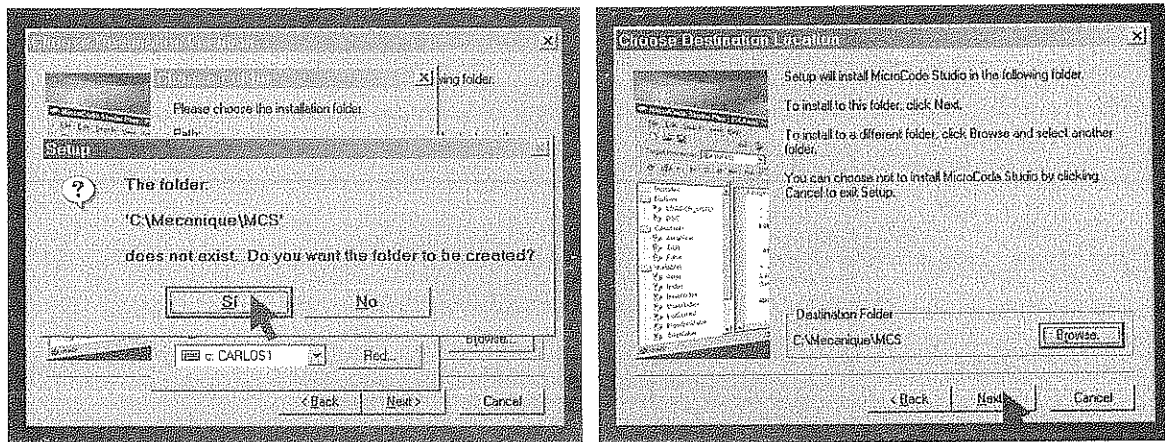
Para instalar nuestro editor primero debemos buscar el archivo mcstudio.exe que terminamos de descargar y lo ejecutamos, veremos una nueva pantalla en la que presionamos next, luego yes.



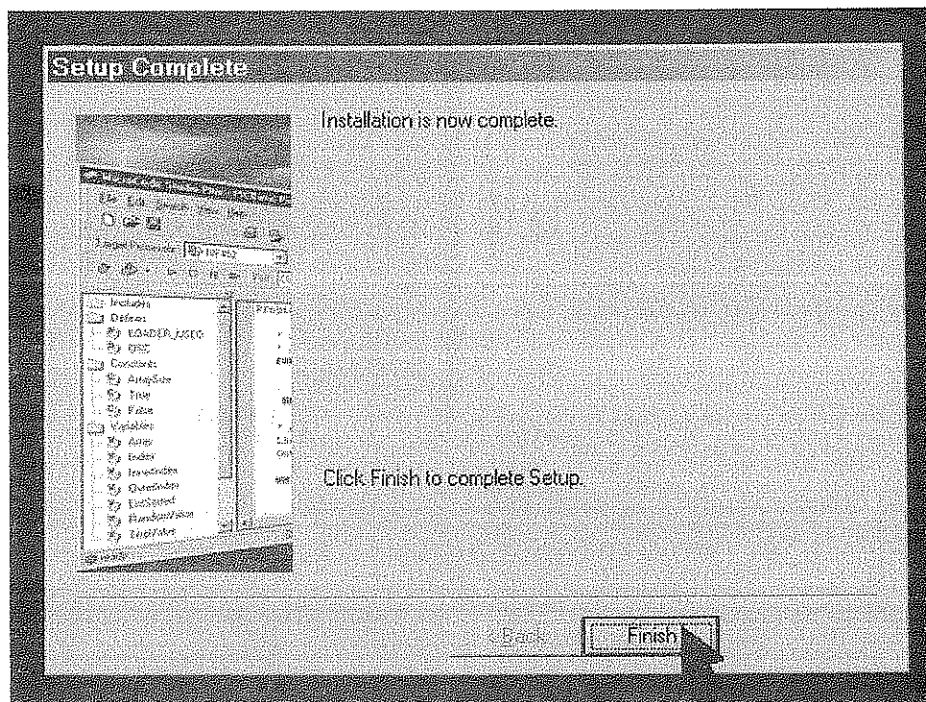
A continuación nos indicará la carpeta en donde se instalará C:\Archivos de Programa\Mecanique si deseamos lo dejamos ahí, pero les recomiendo mejor instalar en C:\ directamente, presionamos Browse ,cambiamos a C:\Mecanique\MCS y presionamos OK.



Inmediatamente preguntará si quiere crear esta nueva ubicación presionamos Si, luego tendremos la pantalla destino de la instalación, debe decir C:\Mecanique\MCS, presionamos Next.



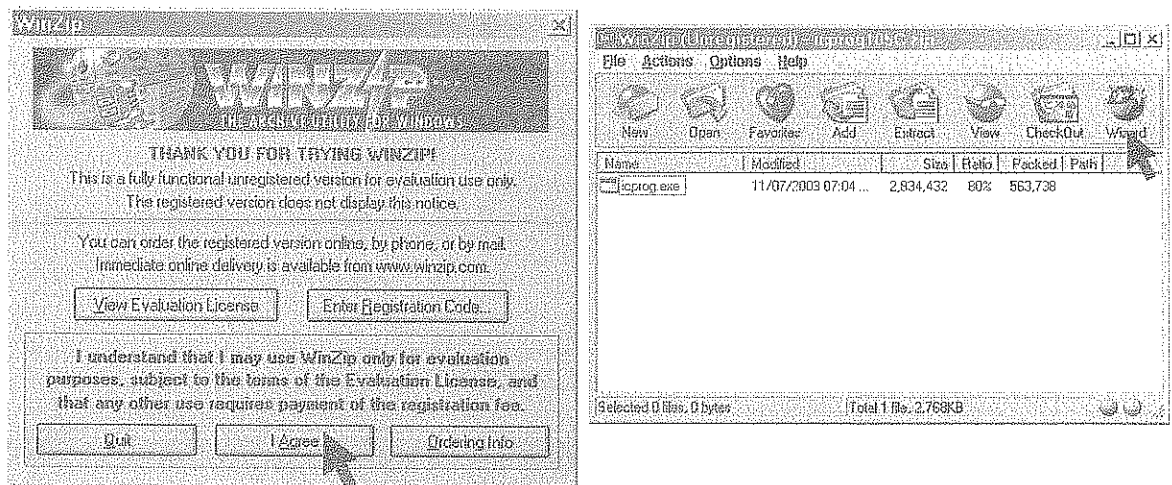
Al cabo de unos segundos la instalación habrá terminado, entonces presionamos Finish.



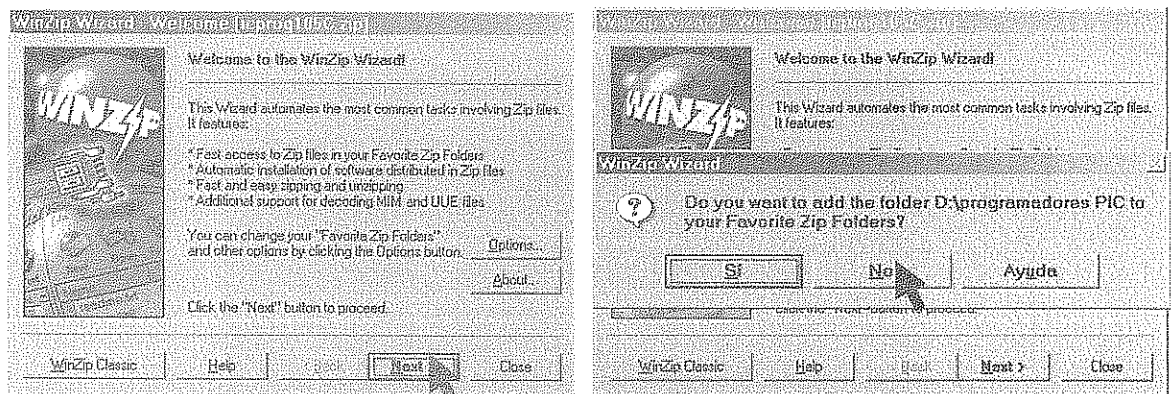
1.5 INSTALACIÓN DEL SOFTWARE PROGRAMADOR IC-Prog 1.05C.

Para instalar este software necesitamos tener instalado **WinZip** en nuestro computador, este es el encargado de descomprimir el archivo **icprog105c.zip**. Una vez localizado el archivo a través del explorador de windows damos doble clic, en ese instante se ejecutará el programa WINZIP, en esta pantalla presionamos la tecla que dice Agregar, solo si nos sale una pantalla similar al de la

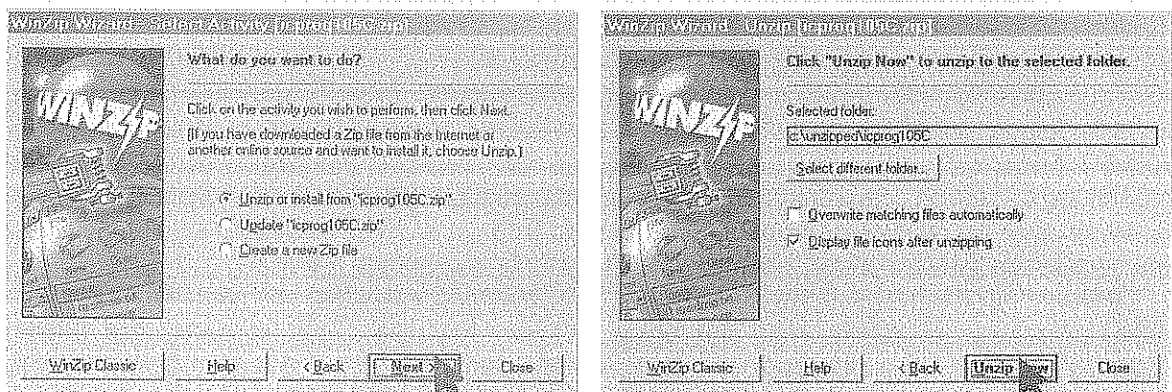
Figura de la derecha, presionamos Wizard.



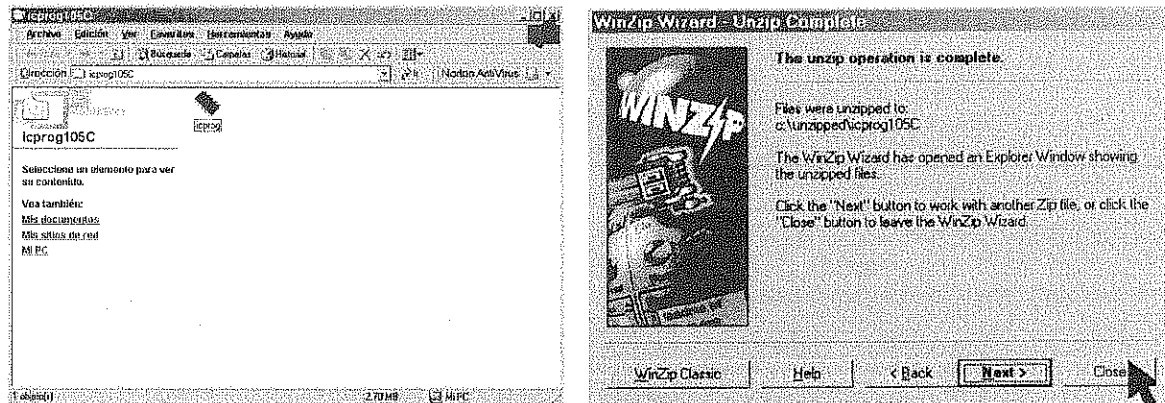
Luego presionamos Next, nos preguntará si queremos adicionar un f6lder favorito ponemos No.



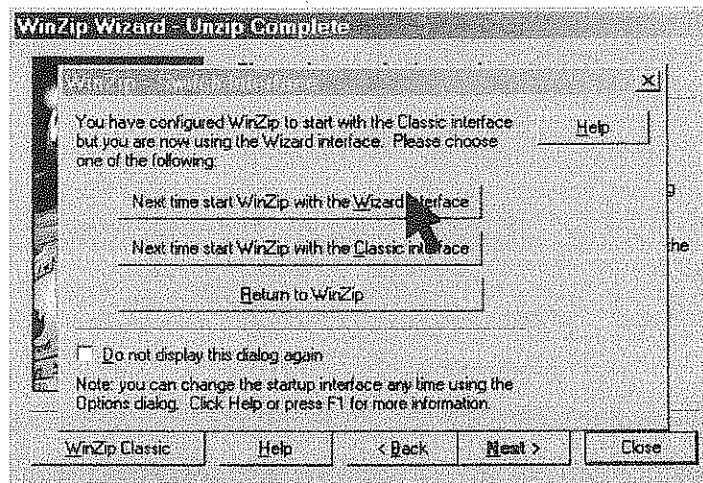
A continuaci3n seleccionamos Unzip or install from "icprog105C.zip" y presionamos Next, luego nos indica la carpeta en donde se va a ubicar lo cual no nos importa porque luego tendremos que reubicarle, en todo caso se instalará en c:\unzipped\icprog105C, presionamos la tecla Unzip Now y esperamos unos segundos.



Aparecerá una ventana de Icprog105C con su ejecutable en el interior, cerramos esta ventana y volvemos a la pantalla de diálogo de WinZip, esta vez presionamos Close.



Inmediatamente aparecerá una pantalla similar al siguiente gráfico, en donde nos preguntará la forma en que desea que se inicie WinZip para las próximas ocasiones, sea modo Classic o Wizard efectivamente presionaremos Wizard, esto nos ahorrará tiempo para la próxima vez que ejecutemos WinZip.



1.6 INSTALACIÓN DEL DRIVER PARA WINDOWS NT/2000/XP.

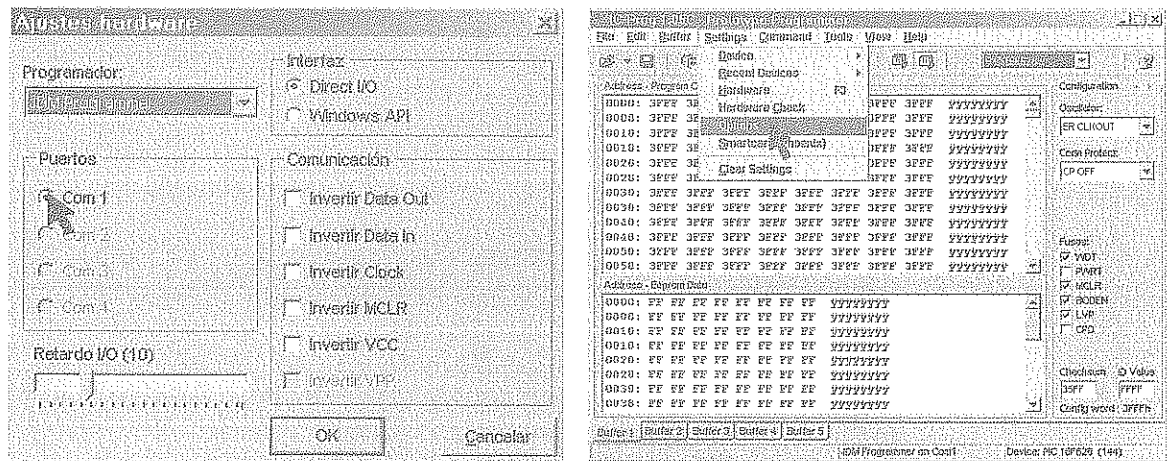
Si usted dispone de cualquiera de estos sistemas operativos es necesario que instale este driver para que funcione correctamente el programador IC-prog105C. Primero necesitamos encontrar el archivo Icprog_driver.zip que descargamos del internet, y procedemos a descomprimir de idéntica forma que lo hicimos en el caso anterior. Una vez que terminemos de descomprimir, tendremos la carpeta icprog_driver, dentro de esta carpeta esta el archivo **icprog.sys**, El cual debemos moverlo dentro de la carpeta Icprog105C, junto al archivo ejecutable icprog.exe, ahora que ya tenemos nuestro driver, tenemos que activarlo de la siguiente manera:

Primero ejecutamos el archivo icprog.exe, la primera vez aparecerá una pantalla en donde se debe seleccionar el puerto con que desea trabajar, hay casos que aparecen dos puertos disponibles, usted debe seleccionar el puerto en donde está conectado el grabador de micros, si no aparece ningún puerto disponible, tiene dos opciones:

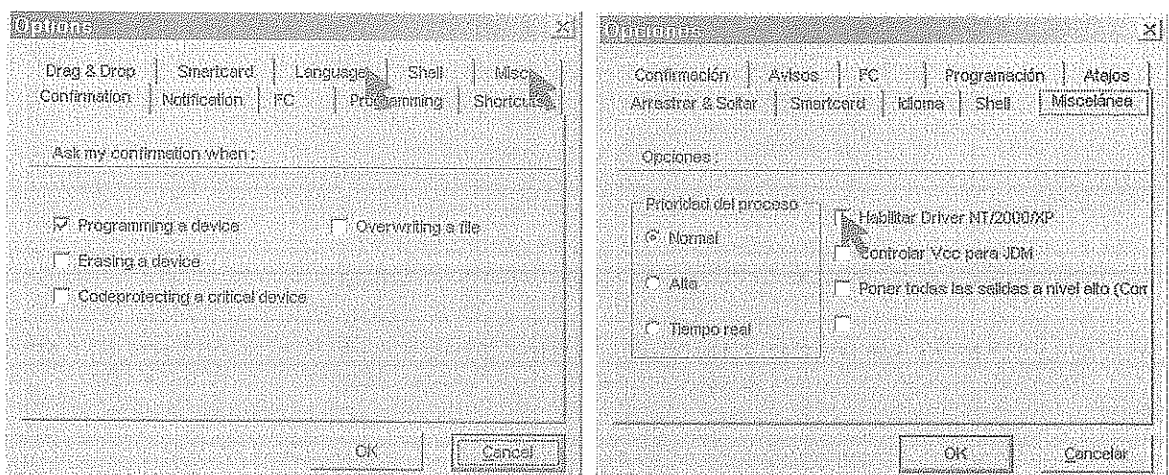
a) Si utiliza un mouse serial y dispone en su computador un conector PS2 es aconsejable comprarse un nuevo mouse el cual conectaremos al PS2 y de esta manera tendremos un puerto serial disponible para nuestro grabador de micros.

b) Si no dispone de ningún puerto serial, lo aconsejable sería comprar una tarjeta de puertos seriales y instalar en su computador.

Una vez solucionado el problema de los puertos seriales damos un clic en **OK**, aparecerá una pantalla similar al de la derecha de los dos siguientes gráficos, en este abrimos **Settings** y luego damos un clic en **Options**.



En esta nueva pantalla, podremos cambiar el lenguaje, lo cual les recomendaría que hicieran primero, y luego presionen **OK**, en ese mismo instante observarán que todo esta en español, bien ahora vuelvan a abrir **Ajustes** (antes llamado **Settings**) y luego den un clic en **Opciones**, esta vez den un clic en **miscelánea** para habilitar el driver de **Windows NT/2000/XP**, una vez que se marque el casillero presionamos **OK** y listo.



1.7 INSTALACIÓN DE pbp244 (PicBasic Pro versión 2.44).

Este programa debemos descomprimirlo de igual forma que aprendimos antes, si deciden utilizar la versión demo de pbp 2.44 deben tener en cuenta que solo puede compilar 30 líneas de

programa, para este libro asumimos que usted tiene la versión completa de pbp 2.44, de todas maneras la mayoría de las prácticas a realizarse, es posible compilar con la versión demo de PICBasic PRO, y además si no dispone de ningún compilador, en el CD se incluye todos los archivos . HEX, necesarios para grabar en el PIC y ponerlos a funcionar.

CAPÍTULO 2

EL MICROCONTROLADOR PIC

2. ¿QUÉ ES UN MICROCONTROLADOR?

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado para que realice desde un simple parpadeo de un led hasta una sofisticada automatización de una fábrica. Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, NAND, conversores A/D, D/A, temporizadores, decodificadores, etc, simplificando todo el diseño a una placa de reducido tamaño y pocos elementos.

2.1. EL MICROCONTROLADOR PIC16F62X (16F627 16F628).

Los microcontroladores PIC (Peripheral interface Controller), son fabricados por la empresa MICROCHIP Technology INC. cuya central se encuentra en Chandler, Arizona, esta empresa se mantiene a la cabeza frente a los demás competidores debido a la gran variedad, gran velocidad, bajos costos, bajo consumo de potencia, y gran disponibilidad de herramientas para su programación.

Uno de los microcontroladores más populares en la actualidad es el PIC16F628 o el PIC16F627, soporta 1000 ciclos de escritura en su memoria FLASH, y 1'000.000 ciclos en su memoria Eeprom, este está reemplazando rápidamente al popular PIC16F84A, pues presenta grandes ventajas frente a este último, como son:

	PIC16F84A	PIC16F627	PIC16F628
Memoria de programa	1024	1024	2048
Memoria datos EEPROM	64	128	128
Memoria RAM	68	224	224
Pines de entrada/salida	13	16	16
Comparadores	-	2	2

Figura 2.1.1. Tabla de comparación entre el PIC16F84A y el PIC16F62X.

Todas estas y otras ventajas más como oscilador interno RC de 4MHZ, MCLR programable, mayor resistencia, comunicación AUSART, etc. Lo hacen al PIC16F62X, como el microcontrolador ideal para estudiantes y aficionados, ya que al tener oscilador interno y el

MCLR (master clear) sea opcional, es mucho más sencillo ponerlo en funcionamiento, basta con conectar al pin 14 a 5V y el pin 5 a tierra para que empiece a trabajar.

-El voltaje de alimentación del PIC16F62X es desde 3 V. hasta 5.5 V. como máximo.
 -sus 2 puertos el A y el B entregan un total de 200mA cada uno, es decir 25 mA cada pin
 -En modo sumidero pueden soportar cada uno de sus puertos 200mA. Es decir 25 mA. cada pin

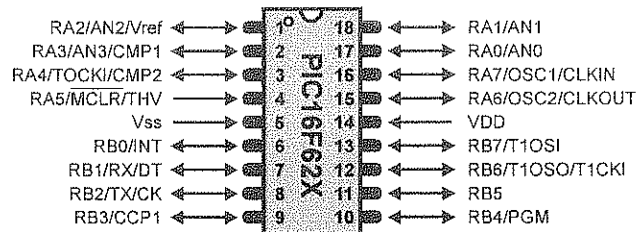
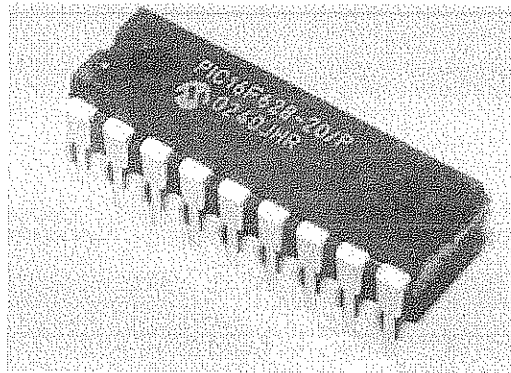


Figura 2.1.2. Presentación más popular del PIC16F62X el PDIP y su diagrama de pines.

2.2. CONSIDERACIONES BÁSICAS PERO MUY ÚTILES A LA HORA DE MONTAR UN PROYECTO.

Es muy importante tomar en cuenta estas recomendaciones ya que si no se las sigue podríamos correr el riesgo de dañar el PIC:

1. recuerde que el PIC tiene tecnología CMOS, esto quiere decir que consume muy poca corriente pero que a la vez es susceptible a daños por estática, se recomienda utilizar pinzas para manipular y así poder transportar desde el grabador al protoboard o viceversa, o a su vez utilizar una manilla antiestática.
2. procure utilizar un regulador de voltaje como el 7805 que nos entrega exactamente 5V. y no un adaptador de pared, ya que el voltaje de salida no siempre es el mismo del que indica su fabricante, por último puede utilizar un circuito con un diodo zener de 5.1 V.
3. no sobrepase los niveles de corriente, tanto de entrada como de salida, recuerde que el PIC puede entregar por cada uno de sus pines una corriente máxima de 25 mA. A si mismo soporta una corriente máxima de entrada de 25 mA., esto quiere decir que puede encender un led con una resistencia de 330 Ω , revisemos:

Voltaje que sale de un pin del PIC, si es alimentado con 5 V. : 5V.

Corriente que requiere el led para un encendido normal: 15 mA.

Calcular la resistencia necesaria.

$$V=R \cdot I \quad R= \frac{V}{I} \quad R= \frac{5 \text{ V.}}{0.015 \text{ A.}} \quad R= 333.33 \Omega \approx 330 \Omega$$

Ahora un ejemplo de corriente de entrada al PIC, si queremos poner un pulsador ¿Qué resistencia sería la mínima a colocarse?

Como sabemos la corriente de entrada que soporta por cada pin del PIC es de 25mA entonces para un pulsador tenemos que:

$$V=R \cdot I \quad R= \frac{V}{I} \quad R= \frac{5 \text{ V.}}{0.025 \text{ A.}} \quad R= 200 \, \Omega \approx 220 \, \Omega$$

Esto quiere decir que la resistencia mínima a colocarse sería de 220 Ω para estar al límite de la capacidad que soporta el PIC, pero no es muy aconsejable trabajar con los límites, por lo que se recomienda utilizar una resistencia de 1 K Ω a 10 K Ω , así el PIC estaría trabajando tranquilamente con una corriente de entrada de 5 mA o 0,5 mA respectivamente.

4. En algunos proyectos es necesario conectar un capacitor de 0,1 μF en paralelo al PIC, este evita mal funcionamientos que podrían ocurrirle, en especial cuando utilizamos teclados matriciales y tenemos adicionalmente un buzzer activo (parlante activo o chicharra).

5. Cuando se necesite precisión en el trabajo del PIC (comunicación serial, tonos DTMF), se recomienda utilizar un cristal oscilador externo de 4 MHz en adelante, ya que el oscilador interno RC que posee no tiene muy buena precisión. En un experimento realizado se conectó dos PIC idénticos con el mismo programa el cual consistía en hacer parpadear un led con intervalos de 1 segundo, ambos PIC compartían la misma fuente y al momento de arrancar los dos parpadeos eran iguales, al transcurso de unos minutos los leds se habían desigualado, esto demuestra que la calibración interna no es igual en todos los micros, si utilizáramos cristales externos de 4 MHz en ambos PIC, no se desigualan nunca, esto debido a que los cristales son muy precisos en cuanto a la frecuencia que entregan.

Con todas estas consideraciones tendríamos el siguiente diagrama para encender un led y conectar un pulsador sin que el PIC sufra ningún daño.

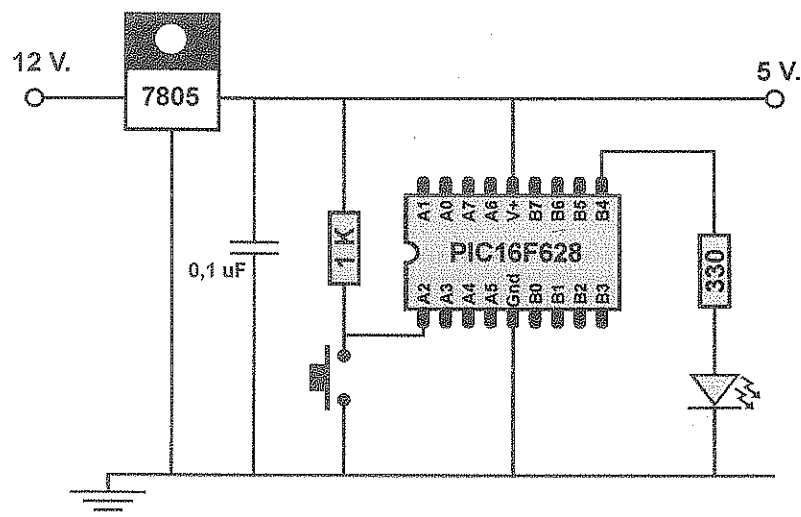


Figura 2.2.1. Diagrama básico para conectar un PIC con un LED y un pulsador, noten que el PIC no necesita oscilador externo ni tampoco resistencia pull-up conectado al MCLR o puerto A5 como lo necesitaba su antecesor el PIC16F84A.

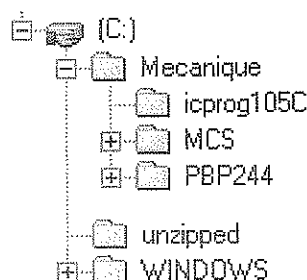
CAPÍTULO 3

EL PROGRAMA MicroCode Studio

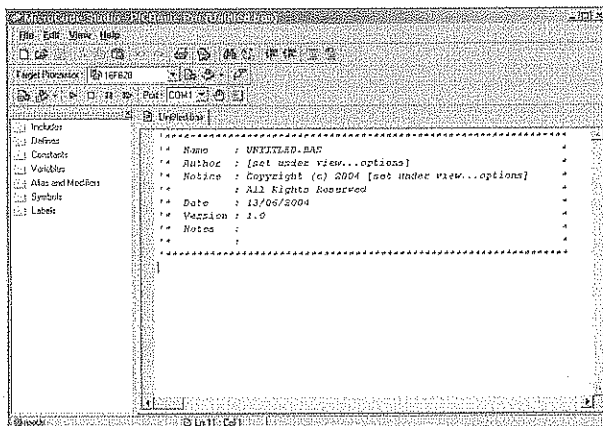
3.1 CONFIGURACIÓN DE MicroCode Studio (MECANIQUE).

En este Capítulo aprenderemos a configurar nuestro editor para tener el mejor rendimiento posible, lo primero que debemos hacer es agrupar dentro de la carpeta de mecanique los dos programas, el pbp 2.44 y el icprog105C, con la finalidad de que la primera vez que ejecutemos el programa microcode, pueda encontrar inmediatamente su compilador pbp 2.44 y su programador IC-prog 105C. Para esto debemos utilizar el explorador de windows y buscar las dos carpetas que seguramente estarán dentro de C:\unzipped, y moverlos o cortar y pegar dentro de C:\mecanique\, es necesario eliminar de su ubicación original, el siguiente gráfico muestra la forma de cómo debería quedar:

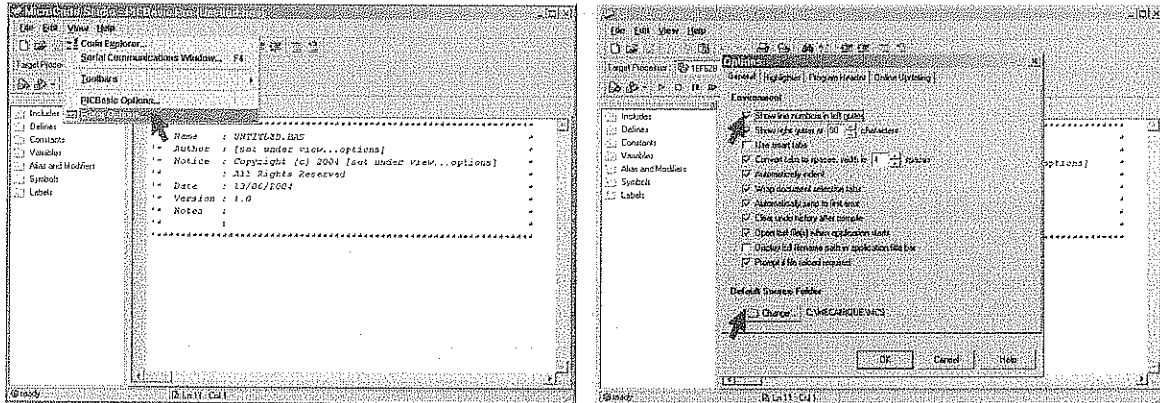
Figura 3.1.1. esquema de la ubicación de cada una de las carpetas



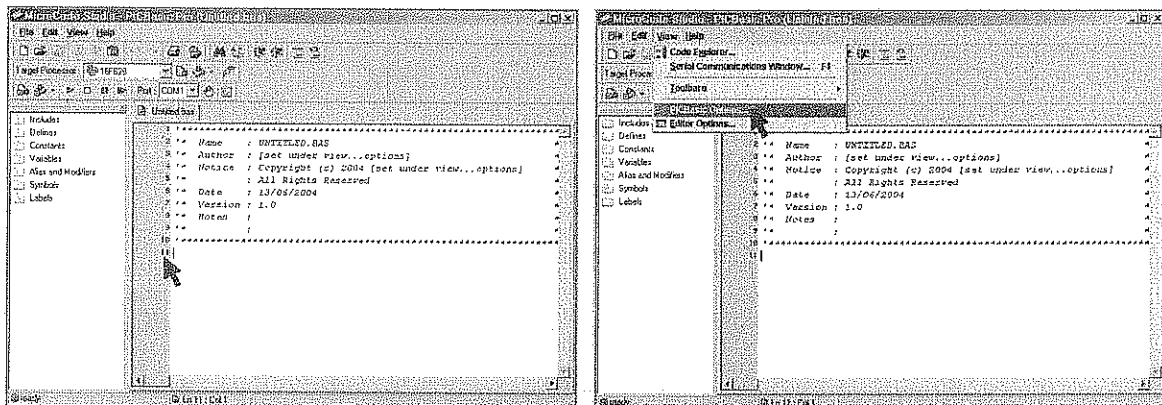
Bien con todas las carpetas juntas vamos a crear en nuestro escritorio un acceso directo del archivo ejecutable C:\mecanique\MCS\CodeStudio.exe, y si lo deseamos también podemos crear un acceso directo de icprog.exe, ahora ejecutamos **CodeStudio.exe** por primera vez, el programa buscará el compilador disponible y tendremos una pantalla similar al siguiente gráfico:



En esta pantalla buscamos **View** y damos clic en **Editor Options...**, luego marcamos **show line numbers in left gutter** que sirve para que aparezca el número de la línea que estamos programando, esto es muy útil al momento de encontrar errores. Si deseamos también podemos indicar la carpeta predefinida en la que queremos guardar nuestros archivos *.bas, por ejemplo podría ser una ubicación donde ya previamente hayamos creado como: C:\mis documentos\ejercicios PIC, si no deseamos crear ningún vínculo solo presionamos la tecla que dice OK.



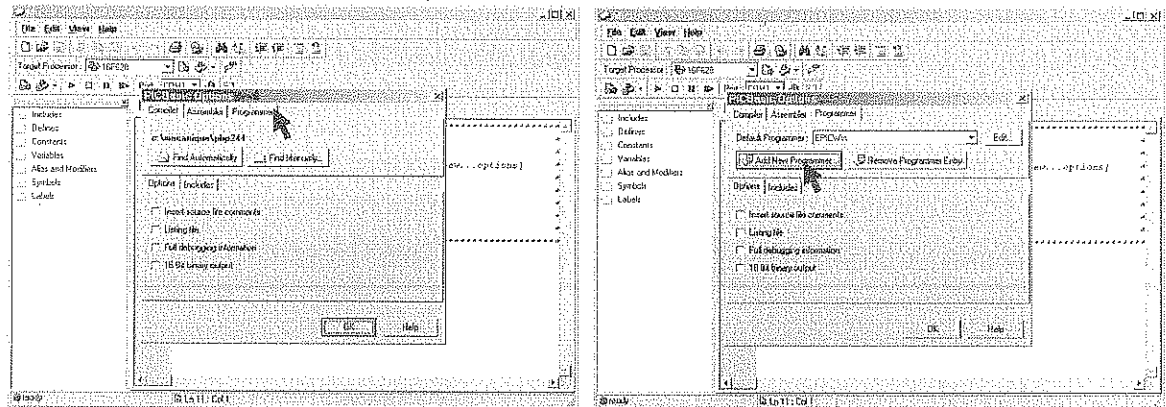
Bien ahora noten que aparecen números al lado izquierdo del editor de texto esto será muy útil en lo posterior, vamos nuevamente a la parte superior donde dice **View** y damos un clic en **PICBasic Options...**



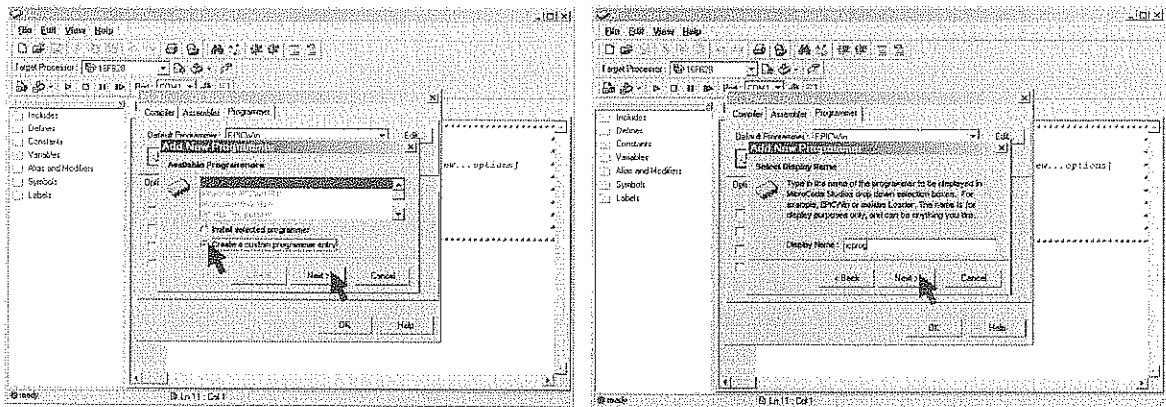
Aparecerá una nueva pantalla más pequeña, en donde vemos que ya está predefinido la ubicación C:\mecanique\pbp244, si apareciera C:\Unzipped\pbp244 es porque no lo borramos de la ubicación anterior, y microcode tomo el pbp244 de este lugar, para corregirlo debemos eliminar el archivo C:\Unzipped\pbp244, y volver a ejecutar el programa microcode, automáticamente lo buscará el archivo pbp 2.44.

Una vez corregido lo anterior presionamos la pestaña Programmer, aquí aparecen los programadores disponibles entre ellos está MicroCode Loader, EPICWin y melabs Loader, pero no aparece Icprog, así que debemos crearlo con la finalidad de que a futuro podamos llamarlo desde microcode, si no lo hacen deberán ejecutar por separado el programa IC-prog creando una demora en la programación.

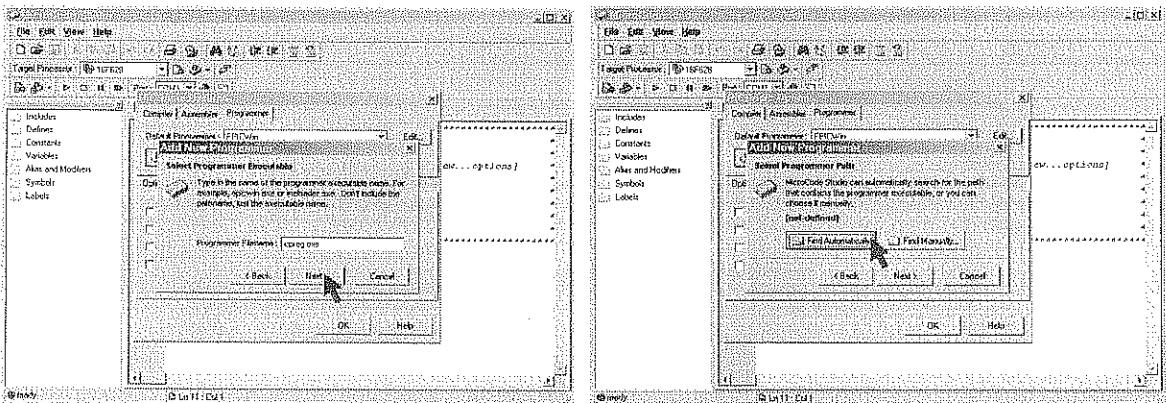
La forma de adicionar el programa IC-Prog es presionando el botón que dice Add New Programmer ver los siguientes gráficos:



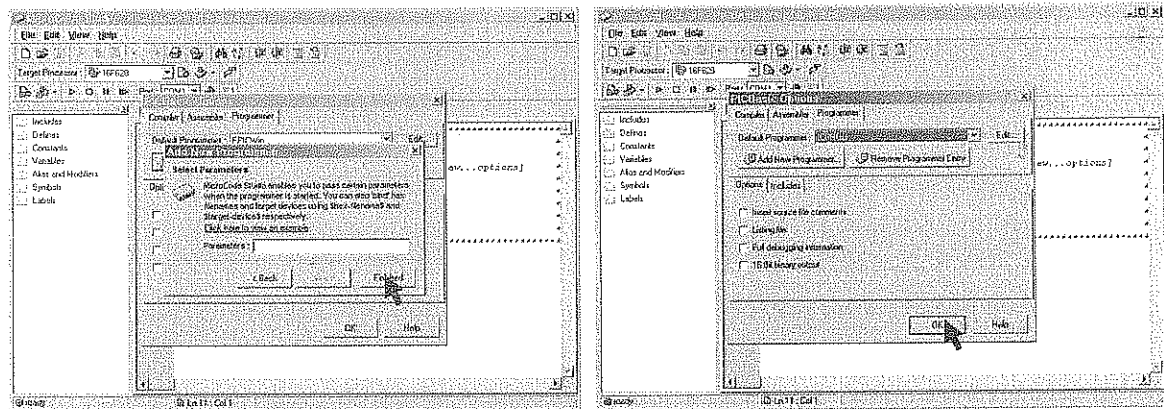
Inmediatamente aparece otra pantalla aun más pequeña en donde marcamos **create a custom programmer entry**, y luego presionamos **Next**, en la siguiente pantalla escribimos cualquier nombre que queremos darle al programador, escribimos icprog y luego presionamos la tecla que dice **Next**.



En esta pantalla nos pedirá que pongamos el nombre del archivo ejecutable escriban: **icprog.exe** y luego presionamos **Next**. Aparecerá otra pantalla con dos botones el uno es para que busque automáticamente la carpeta en donde se encuentra el ejecutable, y el otro para buscar manualmente, como estamos seguros que es el único archivo **icprog.exe** podemos presionar la tecla que dice **Find Automatically**.



En esta pantalla nos pide parámetros, pero para icprog no hace falta así que no escribimos nada y solo presionamos la tecla **Finished**, luego desaparece esta pantalla y solo queda la pantalla de **PICBasic Options**, en donde nos aseguramos de los cambios presionando **OK**.



NOTA: Todos estos ajustes que acabamos de ver se necesitan realizar una sola vez.

3.2 MANEJO DE MicroCode Studio.

Microcode es un programa editor de texto como el Bloc de notas de windows, pero con la diferencia que este esta hecho exclusivamente para facilitar la programación de los microcontroladores PIC, los procedimientos para programar son muy sencillos, primero seleccionamos el modelo del PIC, escribimos el programa y lo guardamos bajo un nombre, en este caso lo guardamos como **led intermitente** y por último presionamos el botón compilar, si el programa está bien escrito y sin fallas compilará y mostrará en la parte inferior izquierda el espacio que requiere en el PIC, y enseguida se creará automáticamente 3 archivos: **led intermitente.mac**, **led intermitente.asm** y **led intermitente.hex**, este último es el más importante para el PIC y es el que grabaremos dentro del microcontrolador, a continuación las partes más importantes de la pantalla de MicroCode Studio:

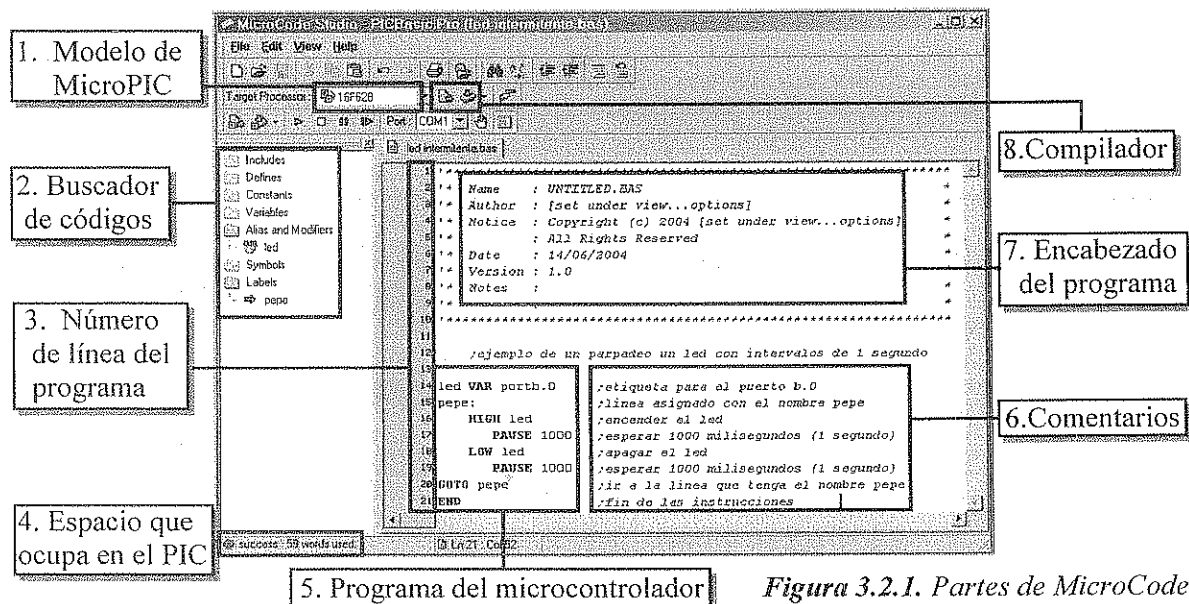


Figura 3.2.1. Partes de MicroCode

1. Modelo de MicroPIC.-

Esto es lo primero que debemos seleccionar antes de empezar a programar, seleccionamos de acuerdo al modelo de Pic que vamos a programar sea este 16F627, 16F628, 16F818, 16F819, 16F84A, 16F877A, etc.

2. Buscador de códigos.-

Aquí se van adicionando cada que creamos una variable, incluimos un define, o creamos algunos nombres de línea, sirve para saber qué componentes incluyen en el programa y también como buscador de líneas, para esto basta con dar un clic en el nombre de la línea que deseamos encontrar, y automáticamente nos indicará donde está dicha línea.

3. Número de línea del programa.-

Esto por defecto no viene habilitado, debemos habilitarlo previamente, y es muy útil a la hora de encontrar errores, porque nos dice el número de la línea en donde se halla un error.

NOTA: Para habilitar esta opción refiérase al Capítulo 3 página 18.

Este no es el tamaño en líneas que ocupa el PIC, sino el que ocupa en Basic.

4. Espacio que ocupa en el PIC.-

Este sí es el espacio que requiere en el Pic y aparece una vez que se compila el programa, debemos fijarnos si alcanza en el PIC que disponemos o debemos reemplazar por otro de mayor capacidad.

NOTA: Para el PIC 16F628 tenemos un espacio disponible de 2048 palabras.

El PIC 16F874 tiene un espacio disponible de 4096 palabras.

El PIC 16F877A tiene un espacio disponible de 8192 palabras.

5. Programa del microcontrolador.-

En esta parte es donde empezamos a escribir nuestro programa, Microcode reconoce palabras clave como **VAR, HIGH, LOW, PAUSE**, etc., y los pinta con mayúsculas y negrillas, por lo que no debemos utilizar estas palabras como nombres de subrutinas o variables.

A continuación interpretamos el significado de cada una de las líneas del siguiente programa, cuyo objetivo es hacer parpadear un led con intervalos de 1 segundo.

Línea 14: led VAR portb.0, indica que el Pin # 6 del PIC 16F628 se llamará en adelante led

Línea 15: pepe:, estamos asignando una subrutina con el nombre de pepe y se lo crea escribiendo cualquier nombre seguido de 2 puntos (:) ejemplo:

Luis:, LUIS:, LuIS:, Alarma:, LedApagado:, Zona3:, Contador:.

NOTA: No podemos empezar con números y tampoco debe contener espacios. ejemplos de lo que no se debe hacer:

3pepe: en su lugar podemos poner pepe3:, pepe 3: el espacio no acepta PICBasic Pro, tampoco acepta pepe3 :, porque hay un espacio entre el 3 y los dos puntos.

Línea 16: HIGH led, significa sacar 5 voltios por el pin 6, lo cual encendería el led.

Línea 17: PAUSE 1000, genera una pausa o retardo de 1000 milisegundos, que equivale a 1s.

NOTA: Los PAUSE que podemos utilizar es de 1 a 65535, es decir que PAUSE 65535, equivale a más de 1 minuto y 5.5 segundos, y PAUSE 1 equivale a 0,001 segundo.

Línea 18: LOW led, significa poner el pin 6 a un estado bajo o 0 voltios, esto apagará el led.

Línea 19: PAUSE 1000, como ya explicamos genera una espera sin hacer nada de 1 segundo.

Línea 20: GOTO pepe, Como el ingles lo dice ir a pepe, indica continuar desde la línea 15, con esto se repetiría el parpadeo del led para siempre.

NOTA: PICBasic Pro ejecuta las instrucciones en orden desde arriba hacia abajo, en el caso del ejercicio anterior desde la línea 14, luego la 15, 16, 17, 18, 19, 20, luego de esta última salta a la línea 14, y nuevamente repite el proceso.

Línea 21: END, Fin de las instrucciones, sirve para indicarle al compilador pbp que hasta aquí es el programa válido.

6. Comentarios.-

Es recomendable usar comentarios todo el tiempo, aunque sea obvio para usted, alguien podría necesitarlo, y porque no para usted mismo, dentro de un tiempo no recordará ni como lo hizo ni como funciona, ni para qué servía tal instrucción.

NOTA: Los comentarios se crean anteponiendo un punto y coma (;), noten que el texto cambia de color de negro a azul y del tipo cursiva.

Trate de poner comentarios entendibles por ejemplo:

HIGH portb.3 ;activar el relé, que enciende el MOTOR.

7. Encabezado del programa.-

No son nada más que comentarios en los que podemos incluir nombre, fecha, autor, y una explicación en breves palabras de cómo y para qué sirve el programa. También podemos hacerlo modificando en **View ---Editor Options---Program header**, aquí colocamos el autor y empresa para que se coloque automáticamente cada que abrimos una página nueva.

8. Compilador.-

Estos 2 botones sirven básicamente para compilar nuestro programa y crear el archivo .ASM , .MAC , y el .HEX, el .HEX sirve para grabar en el micro, el .MAC solo sirve para el PICBasic y el .ASM, para personas interesadas en ver como lo hizo el compilador en assembler ya que podemos abrirlo en MPLAB.



Compile Only - F9 .Este primer botón sirve para compilar, es decir nuestro programa lo cambia a assembler y lo crea el .HEX, más adelante veremos como trabaja.



Compile and Program - F10. Este botón tiene doble función , aparte de hacer lo mismo que el botón anterior , es decir compilar , también puede llamar al programador Ic-prog , con la finalidad de ahorrarnos tiempo y no tener que abrir por separado, es aconsejable utilizarlo una sola vez , y una vez que el programador ICProg ya está abierto , en adelante solo utilizaremos el botón **Compile Only - F9**.

3.3 DECLARACIONES DISPONIBLES EN EL COMPILADOR PBP 2.44.

Debemos entender que declaraciones son cada una de las palabras que el compilador pbp 2.44 tiene reservado para realizar una tarea específica, las más utilizadas son: **HIGH**, **LOW**, **PAUSE**, **GOSUB**, **GOTO**, **LCDOUT**, **SERIN**, **SEROUT**, **FOR**, **NEXT**, **IF**, **THEN**, **SOUND**, **END**, un ejemplo:

HIGH portb.3

Esta instrucción **HIGH** es reconocida automáticamente por microcode, lo coloca en negrilla y mayúscula, y sirve para que el compilador realice los ajustes necesarios para cambiarse al segundo banco de la RAM, coloque como salida el puerto B en TRISB, y luego regrese al primer banco de la RAM y setea en 1 al Portb.3, todo esto nos ahorramos gracias al pbp 2.44.

A continuación las 83 instrucciones disponibles con una breve explicación.

DECLARACIÓN	APLICACIÓN
@	Inserta una línea de código ensamblador
ADCIN	Lee el conversor analógico
ASM...ENDASM	Insertar una sección de código ensamblador
BRANCH	GOTO computado (equivale a ON..GOTO)
BRANCHL	BRANCH fuera de página (BRANCH Largo)
BUTTON	Anti-rebote y auto-repetición de entrada en el pin especificado
CALL	Llamada a subrutina de ensamblador
CLEAR	Hace cero todas las variables
CLEARWDT	Hace cero el contador del Watchdog Timer
COUNT	Cuenta el número de pulsos en un pin
DATA	Define el contenido inicial en un chip EEPROM
DEBUG	Señal asincrónica de salida en un pin fijo y baud
DEBUGIN	Señal asincrónica de entrada en un pin fijo y baud
DISABLE	Deshabilita el procesamiento de ON INTERRUPT, ON DEBUG
DISABLE DEBUG	Deshabilita el procesamiento de ON DEBUG
DISABLE INTERRUPT	Deshabilita el procesamiento de ON INTERRUPT
DTMFOUT	Produce tonos telefónicos en un pin
EEPROM	Define el contenido inicial en un chip EEPROM
ENABLE	Habilita el procesamiento de ON INTERRUPT, ON DEBUG
ENABLE DEBUG	Habilita el procesamiento de ON DEBUG
ENABLE INTERRUPT	Habilita el procesamiento de ON INTERRUPT
END	Detiene la ejecución e ingresa en modo de baja potencia
FOR...NEXT	Ejecuta declaraciones en forma repetitiva
FREQOUT	Produce hasta 2 frecuencias en un pin
GOSUB	Llama a una subrutina BASIC en la línea especificada
GOTO	Continúa la ejecución en la línea especificada
HIGH	Saca un 1 lógico (5 V.) por un pin
HPWM	Salida de hardware con ancho de pulsos modulados
HSERIN	Entrada serial asincrónica (hardware)
HSEROUT	Salida serial asincrónica (hardware)
I2CREAD	Lee bytes de dispositivos I2C
I2CWRITE	Graba bytes de dispositivos I2C
IF..THEN..ELSE..ENDIF	Ejecuta declaraciones en forma condicional
INPUT	Convierte un pin en entrada
LCDIN	Lee caracteres desde una RAM de un LCD
LCDOUT	Muestra caracteres en un LCD

LET	Asigna el resultado de una expresión a una variable
LOOKDOWN	Busca un valor en una tabla de constantes
LOOKDOWN2	Busca un valor en una tabla de constantes o variables
LOOKUP	Obtiene un valor constante de una tabla
LOOKUP2	Obtiene un valor constante o variable de una tabla
LOW	Hace 0 lógico (0 V.) un pin específico
NAP	Apaga el procesador por un corto periodo de tiempo
ON DEBUG	Ejecuta un Debug en BASIC
ON INTERRUPT	Ejecuta una subrutina BASIC en un interrupt
OUTPUT	Convierte un pin en salida
OWIN	Entrada de dispositivos un-alambre
OWOUT	Salida a dispositivos un-alambre
PAUSE	Demora con resolución de 1 milisegundo
PAUSEUS	Demora con resolución de 1 microsegundo
PEEK	Lee un byte del registro
POKE	Graba un byte en el registro
POT	Lee el potenciómetro en el pin especificado
PULSIN	Mide el ancho de pulso en un pin
PULSOUT	Genera pulso hacia un pin
PWM	Salida modulada en ancho de pulso por un pin especificado
RANDOM	Genera número pseudo-aleatorio
RCTIME	Mide el ancho de pulso en un pin
READ	Lee byte de un chip EEPROM
READCODE	Lee palabra desde un código de memoria
RESUME	Continúa la ejecución después de una interrupción
RETURN	Continúa en la declaración que sigue al último GOSUB
REVERSE	Convierte un pin de salida en entrada, o uno de entrada en salida
SELECT CASE	Compara una variable con diferentes valores
SERIN	Entrada serial asincrónica (tipo BASIC Stamp 1)
SERIN2	Entrada serial asincrónica (tipo BASIC Stamp 2)
SEROUT	Salida serial asincrónica (tipo BS1)
SEROUT2	Salida serial asincrónica (tipo BS2)
SHIFTIN	Entrada serial sincrónica
SHIFTOUT	Salida serial sincrónica
SLEEP	Apaga el procesador por un periodo de tiempo
SOUND	Genera un tono o ruido blanco en un pin
STOP	Detiene la ejecución del programa
SWAP	intercambia los valores de dos variables
TOGGLE	Hace salida a un pin y cambia su estado
USBIN	Entrada de USB
USBINIT	Inicializar USB
USBOUT	Salida de USB
WHILE...WEND	Ejecuta declaraciones mientras la condición sea cierta
WRITE	Graba bytes en un chip EEPROM
WRITECODE	Escribe palabra en código de memoria
XIN	Entrada X - 10
XOUT	Salida X - 10

NOTA: si desea más información de cada declaración puede ver la ayuda de microcode en Help topics\ Statement reference, o visite la web WWW.frino.com.ar y descargue un manual de php.

3.4 IDENTIFICACIÓN DE ERRORES EN LA COMPILACIÓN.

En el momento que compilamos un programa, este realiza una previa verificación del mismo, si existen errores microcode señala el primer error que encuentra con una franja CAFÉ, luego en la parte inferior menciona los demás errores con el número de línea y su explicación, por eso se recomienda activar la opción que muestra el número de línea de programación, (si desea activar esta opción refiérase al capítulo 3 página 18), a continuación un ejemplo de error en la compilación.

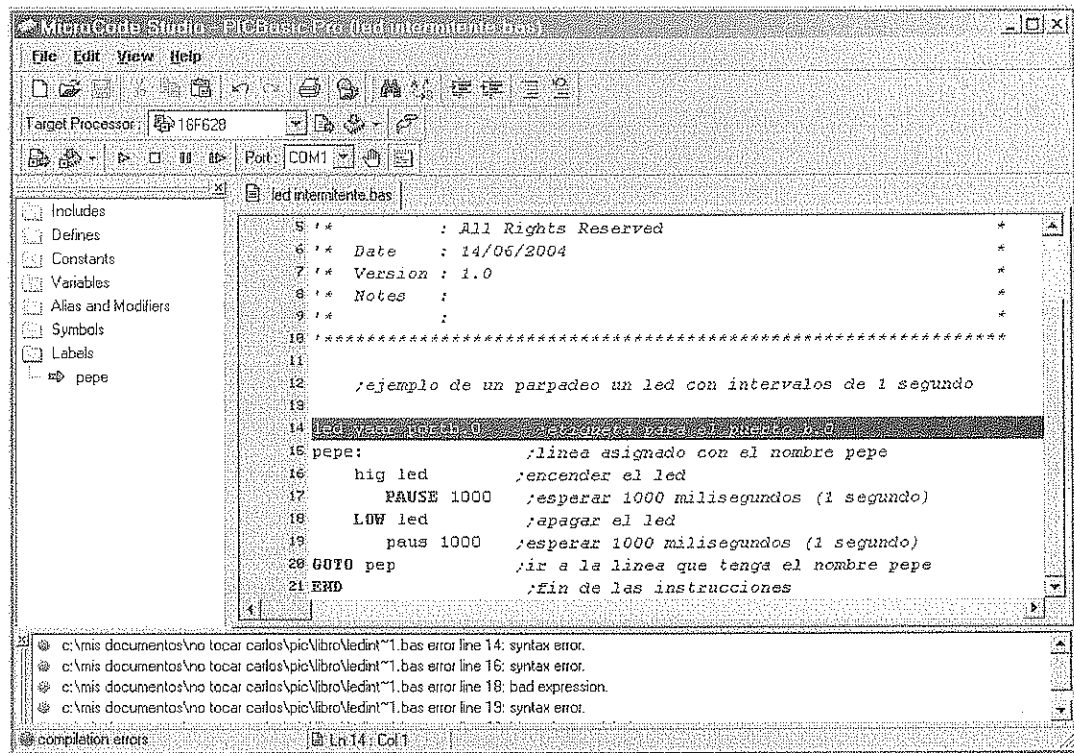


Figura 3.4.1. Pantalla de error en la compilación

Mensaje	Explicación
Syntax error	error de sintaxis, mal escrito o falta o está demás una letra
Bad expresión	mala expresión, mal escrito o falta o está demás una letra
Id pep is not a label	la línea pep no es un nivel, o nombre de línea incorrecto
For without a matching next	cuando falta un next
next without a matching for	cuando falta un FOR ejem. Fo x = 1 to 12
undefined symbol "portc"	cuando ponemos un Puerto que no dispone el pic
80000 numeric overflow	nos excedimos del valor límite ejem. PAUSE 80000
bad token ""	Nos olvidamos poner el número del pin 1,2,3. Ejem. LOW portb.
bad variable modifier: .O.	Ejem. LOW portb.O pusimos la letra o (O) en vez del cero (0)
processor file 12F675	Este error sale en compiladores de versiones antiguas, ya que no dispone de este modelo de PIC por ejemplo en el PBP 2.33
undefined symbol "cncom"	Indica que no existe ese registro en el PIC seleccionado
code crossed boundary @800h	Es una precaución que indica que el programa sobrepasa las 2048 líneas de programación, aunque si compila no es un problema.

Figura 3.4.2. Tabla de los errores más comunes.

CAPÍTULO 4

PROGRAMANDO EN BASIC

4.1 DIFERENCIA ENTRE EL LENGUAJE BASIC Y EL ENSAMBLADOR

Para poder entender la diferencia entre los dos lenguajes de programación, debemos entender qué es un lenguaje de alto nivel y qué es un lenguaje de bajo nivel, a través del siguiente cuadro podemos ver los niveles de programación.

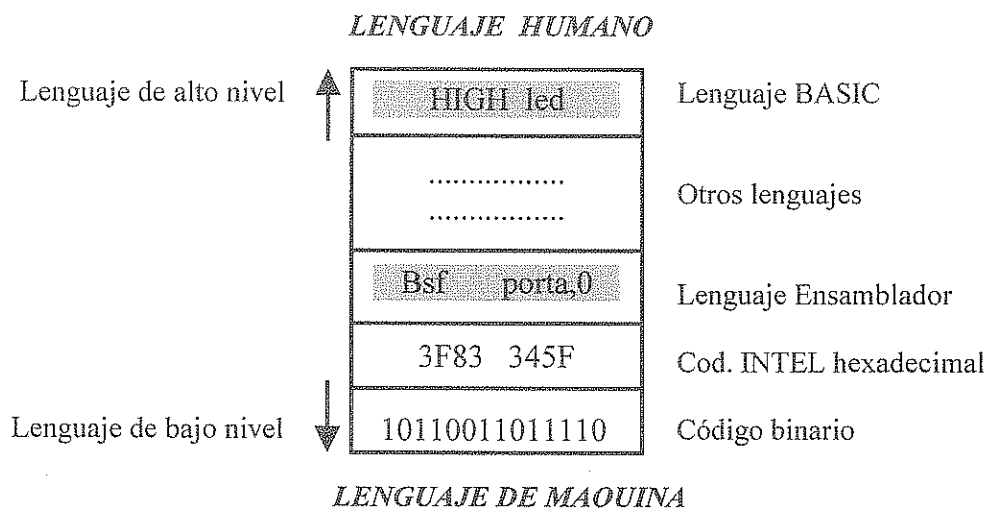


Figura 4.1.1. Cuadro de los niveles de programación, el lenguaje que más se acerca a los humanos es el de más alto nivel, el lenguaje más próximo al tipo de datos que entiende el microcontrolador es un lenguaje de bajo nivel.

Por consiguiente vamos a trabajar con un programador de alto nivel, el que más entendemos los humanos, esta es la gran diferencia entre Ensamblador y BASIC, a continuación un ejemplo de un programa en Basic para el PIC16F628 que hace parpadear un led con intervalos de 1 segundo.

inicio:	; nombre de subrutina inicio
HIGH portb.1	; enciende el led que esta conectado en el pin 7
PAUSE 1000	; espera un segundo
LOW portb.1	; apaga el led
PAUSE 1000	; espera un segundo
GOTO inicio	; continúa el programa para siempre

A continuación el mismo proyecto para el parpadeo del led pero en lenguaje ensamblador.

list p=16F628			
status	equ	03h	;etiquetamos cada posición de memoria
portb	equ	06h	
trisb	equ	86h	
cont1	equ	20h	
cont2	equ	21h	
cont3	equ	22h	
reset	org	0	;se escribe en la línea 0 la instrucción
	goto	inicio	;salta a la línea etiquetada con inicio
	org	5	;las siguientes líneas se escribirán desde la dirección 5
retardo	movlw	D'10'	;El registro cont1 contiene el número de
	movwf	cont1	;veces que repite 100 milisegundos
repitel	movlw	D'100'	;El registro cont2 contiene el número de
	movwf	cont2	;veces que repite 1 milisegundo
repite2	movlw	D'110'	;El registro cont3 contiene el número de
	movwf	cont3	;veces que repite los 9 microsegundos
repite3	nop		;de retardo generados
	nop		;por los 6 ciclos de las instrucciones nop (6usg)
	nop		;más 1 ciclo de la instrucción decfsc (1usg)
	nop		;más 2 ciclos del salto goto (2usg)
	nop		;dando en total los 9usg, siendo esta la base
	nop		;de tiempo, por lo tanto 1sg= 9usg*110*100*10
	decfsz	cont3	;decrementa el reg cont3 y salta si llega a 0
	goto	repite3	;si cont3 no es 0 entonces salta a repite3
	decfsz	cont2	;decrementa el reg cont2 y salta si llega a 0
	goto	repite2	;si cont2 no es 0 entonces salta a repite2
	decfsz	cont1	;decrementa el reg cont1 y salta si llega a 0
	goto	repitel	;si cont1 no es 0 entonces salta a repitel
	retlw	0	;salida de la subrutina cargando w con 0
inicio	bsf	status,5	;se ubica en el segundo banco de la RAM
	movlw	00h	;se carga el registro w con 00h
	movwf	trisb	;se programa el puerto B como salidas
	bcf	status,5	;se ubica en el primer banco de la RAM
prog	bsf	portb,1	;coloca en 1 el pin RB1 para encender el led
	call	retardo	;Llama a la subrutina retardo de 1sg
	bcf	portb,1	;Coloca en 0 el pin RB1 para apagar el led
	call	retardo	;Llama a la subrutina retardo de 1 sg
	goto	prog	;salta a prog para repetir la secuencia
end			

Como podemos ver es mucho más largo y difícil de entender, y además debemos conocer las posiciones de memoria, así como también la arquitectura del PIC, también se debe hacer cálculos muy precisos para generar el retardo de 1 segundo.

También hay que considerar el tiempo que nos demoramos en programar en Assembler con el tiempo que nos demoramos en programar en BASIC, las herramientas que nos facilita el compilador de PicBasic Pro son muy útiles y de gran ahorro de trabajo, un ejemplo es la llamada telefónica que solo con una línea de escritura ya nos genera los tonos DTMF esto es:

DTMFOUT portb.3 , [0,9,7,6,3,8,4,5,6] ;genera tonos telefónicos por el pin B.3

Si bien en esta instrucción nos demoramos 5 segundos en escribir, en ensamblador necesitaremos más o menos 5 horas para escribir las 260 líneas de programa que se calcula que podría tener, además el trabajo de consultar la frecuencia que genera cada una de las teclas DTMF.

En el compilador PicBasic Pro, ya nos facilitan este trabajo, pues los tonos telefónicos están listos solo hay que llamarlos con la declaración **DTMFOUT** y especificar por cual pin vamos a sacar los tonos telefónicos, ejemplos como estos hay muchos.

La única ventaja de programar en Ensamblador es la optimización de espacio en el PIC, si bien es cierto que PicBasic Pro nos ayuda con el tiempo, necesita más espacio que el que ocuparía al programar en Ensamblador, pero esto ya no es un problema si tomamos en cuenta que cada vez fabrican microcontroladores más baratos y con más espacio de memoria.

4.2 APRENDIENDO A PROGRAMAR EL PIC16F628 CON MICROCODE.

Este es el objetivo primordial de este libro, enseñar a programar micros PIC de la forma más rápida posible, si no ha leído completamente el libro le recomendamos leer el literal 3.2 MANEJO DE MicroCode Studio página 20.

Vamos a proponer nuevamente el proyecto que ya hemos visto antes, pero esta vez lo pondremos en funcionamiento, escribimos el programa que viene a continuación, o abrimos el archivo del CD Ejercicios\led intermitente.bas.


```

1 *****
2 ' Name : UNTITLED.EAS
3 ' Author : [set under view...options]
4 ' Notice : Copyright (c) 2004 [set under view...options]
5 ' : All Rights Reserved
6 ' Date : 14/06/2004
7 ' Version : 1.0
8 ' Notes :
9 ' :
10 *****
11
12 :ejemplo de un parpadeo un led con intervalos de 1 segundo
13
14 led VAR portb.0 ;etiqueta para el puerto b.0
15 pepe: ;linea asignado con el nombre pepe
16 HIGH led ;encender el led
17 PAUSE 1000 ;esperar 1000 milisegundos (1 segundo)
18 LOW led ;apagar el led
19 PAUSE 1000 ;esperar 1000 milisegundos (1 segundo)
20 GOTO pepe ;ir a la linea que tenga el nombre pepe
21 END ;fin de las instrucciones

```

Figura 4.2.1. Programa microcode con el archivo led intermitente.bas

1. Recuerde seleccionar el PIC16F628 en Target Processor.

2. Una vez que estemos seguros que el programa esta bien escrito presionamos  o **F9**, observen en la parte inferior izquierda como cambia el color del círculo:



Círculo verde ready, mientras estamos escribiendo el programa.



Círculo rojo mientras está compilando el programa con pbp 2.44.



Círculo amarillo mientras genera el código Assembler y el intel .Hex.




Círculo verde finalizar y nos da el tamaño que necesita en el PIC.

Caso contrario si el programa está mal escrito:



Círculo rojo después de compiling indica error en la compilación.

4.3. GRABANDO EL PIC CON EL IC-Prog 1.05C.

Si todo está bien y nos dice **success : 59 words used.**, presionamos  o **F10**, esperamos a que compile nuevamente y se abra el IC-Prog 1.05C, si es la primera vez que ejecuta este programa no olvide revisar el Capítulo 1 página 10. A continuación la pantalla de IC-Prog 1.05C.

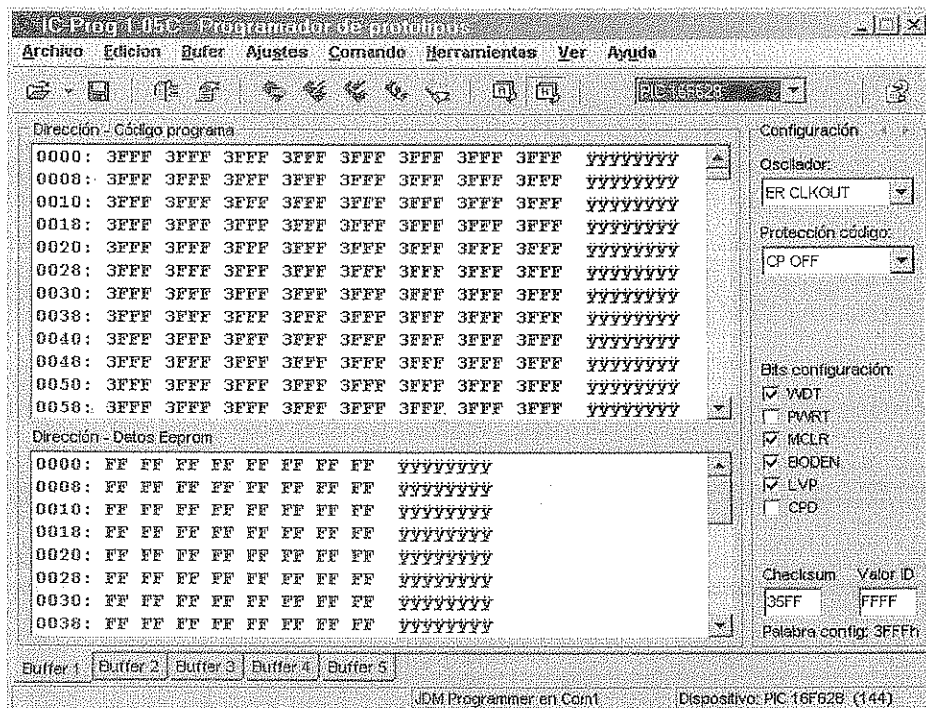


Figura 4.3.1. Presentación de la pantalla de IC-Prog 1.05C.

3. Seleccionar el PIC que se va a grabar, noten que el código de programa esta solo 3FFF 3FFF esto quiere decir que esta vacío, no hay ningún programa a grabarse.

4. Abrimos Archivo y buscamos **led intermitente.HEX**, ponemos Abrir e inmediatamente veremos que el código ha cambiado por algunos números, este es el programa que el PIC entiende, la presentación será similar a la siguiente pantalla:

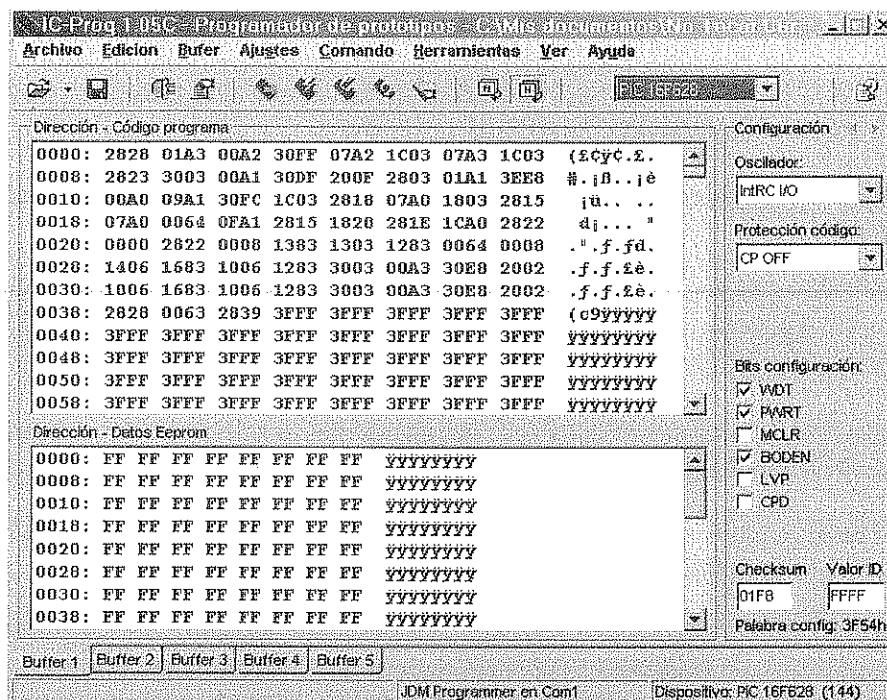



Figura 4.3.2. Pantalla de IC-Prog con el archivo led intermitente.hex cargado.

La Configuración del Oscilador debe estar en **intRC I/O** (Oscilador interno resistencia condensador pin de entrada o salida los A6 y A7), el **MCLR** (reset externo) debe estar deshabilitado, y la protección de código apagada. Si ya está listo e instalado el PIC en el Grabador de micros, presionamos  o F5 y esperamos a que salga el siguiente mensaje:

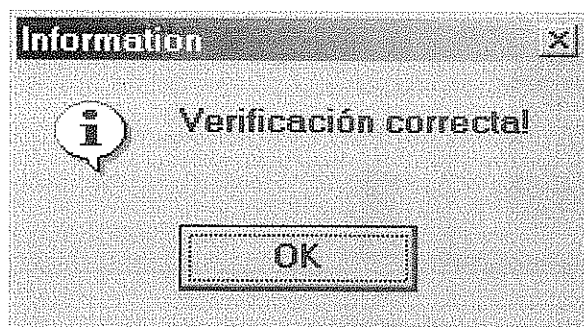


Figura 4.3.3. Cuadro de diálogo que indica que el PIC se grabó exitosamente

En ocasiones puede salir un mensaje similar al siguiente:



Figura 4.3.4. Cuadro de diálogo que indica error en la programación del PIC

Los motivos por los que sale el mensaje error en el código 0000h podrían ser los siguientes:

1. Si el LED rojo del grabador no se encendió mientras estaba programando, revise si está conectado en el puerto com correcto.
2. Si dispone de dos puertos com pruebe cambiando al otro puerto com hasta que se encienda el led rojo del grabador

NOTA: El LED indicador ROJO del grabador solo se enciende mientras se está grabando o leyendo un PIC, es posible que el led verde este encendiéndose, esto no es un problema, simplemente no haga caso.

3. Si el LED rojo del grabador se enciende pero de todas maneras sale el mismo mensaje de error, revise si el PIC está correctamente insertado en el grabador.
4. Si analizado los 3 puntos anteriores sigue saliendo el mismo mensaje, es muy probable que el PIC se encuentre dañado, reemplace por otro e intente grabar nuevamente.

NOTA: Este mensaje de error también sale cuando ponemos grabar un micro con protección en el código de programa, aún cuando el PIC se encuentra en perfectas condiciones, obviamente porque IC-Prog no pudo leer y verificar el contenido del PIC, simplemente en este caso ponga a trabajar el micro y verá que funciona correctamente.

Si ya solucionó el problema y el mensaje es **verificación correcta**, es hora de montar el proyecto y ver funcionar. A continuación conecte como ilustra la siguiente figura y listo.

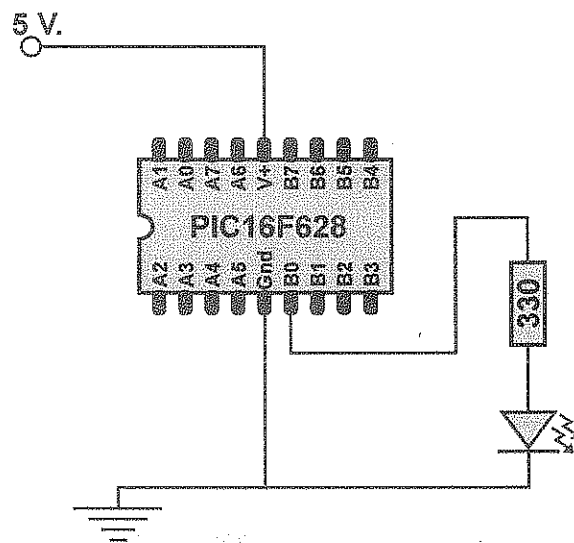


Figura 4.3.5. Conexión de un LED en el puerto B0 (pin 6)

NOTA: si decide utilizar algún pin del puerto A, tome en cuenta que estos son analógicos y podría observar un funcionamiento defectuoso. Para solucionar esto agregue al principio del programa `cmcon= 7`, esto convierte los pines del puerto A en digitales, un inconveniente también es el puerto A5 este es solo de entrada, es decir podemos utilizar para un pulsador pero no para encender un LED, otro inconveniente podría ser el puerto A4 este es de colector abierto, necesita conectarse a 5 voltios, como ilustra la siguiente figura.

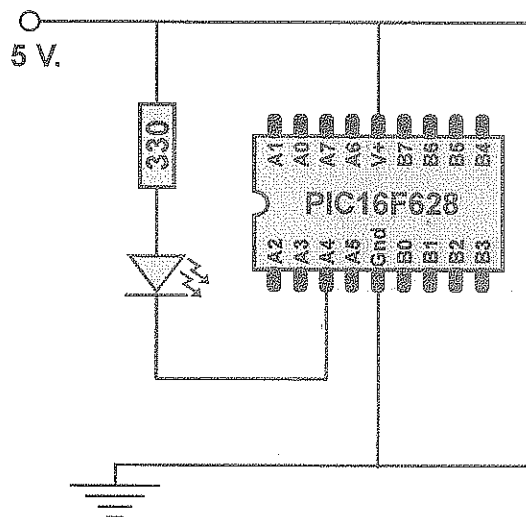


Figura 4.3.6. Diagrama especial de conexión de un LED en el puerto A4 por ser de colector abierto, su lógica es inversa es decir cuando ponemos HIGH se apaga y LOW se enciende

Para los demás puertos A0, A1, A2, A3, A6, A7, estos funcionan normalmente como el puerto B tanto como para entradas o salidas, siempre que incluya la línea cmcon=7.

```
cmcon = 7      ;apaga los comparadores de voltaje del puerto A y los convierte en digitales
Led VAR portb.0      ; etiqueta asignada al pin 17 o Puerto A0
inicio:           ; nombre de subrutina inicio
    HIGH led      ; enciende el led que esta conectado en el pin 17
    PAUSE 1000    ; espera un segundo
    LOW led       ; apaga el led
    PAUSE 1000    ; espera un segundo
    GOTO inicio   ; continúa el programa para siempre
END
```

Figura 4.3.7. Programa de parpadeo de un LED en el puerto A0 convirtiendo en digital A0

4.4. DIFERENTES CAMINOS A SEGUIR PARA CONSEGUIR UN MISMO OBJETIVO.

En este literal se pretende aclarar que existe varias formas de desarrollar un programa, se presentarán a continuación varias maneras de escribir un programa que realiza el mismo trabajo final, es decir hacer parpadear un led con intervalos de 1 segundo en el puerto B0.

```
led VAR portb.0      ; etiqueta asignada al pin 6 o Puerto B0
inicio:             ; nombre de subrutina inicio
    LOW led         ; apaga el led que esta conectado en el pin 6
    PAUSE 1000      ; espera un segundo
    TOGGLE led      ; cambia el estado de portb.0 de on a off o viceversa
    PAUSE 1000      ; espera un segundo
    GOTO inicio     ; continúa el programa para siempre
```

Figura 4.4.1. Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo utilizando TOGGLE que sirve para invertir el estado de un puerto.

En el siguiente caso manejaremos todo el puerto B como salidas, pero solo trabajaremos con una de ellas el puerto B.0, es importante no olvidar incluir al principio del programa `trsb = 0`, o `trsb=%0` ya que sin este no funciona el manejo de puertos, debe entender que `Portb = %00000010` quiere decir encender únicamente el puerto B1 de esta manera tenemos el siguiente ejemplo en donde se explica mejor el manejo del puerto B.

Portb= % 0 1 0 0 1 0 1 0
 B7 B6 B5 B4 B3 B2 B1 B0

Indica encender el Puerto B.6, el B.3 y el B.1, para todos los demás significa permanecer apagados, esto es muy útil en casos en que se va a encender un grupo de leds, como los proyectos que más adelante veremos como el semáforo y luces del auto fantástico, de todas maneras lo emplearemos para hacer parpadear un led.

NOTA: PBP reconoce bases numéricas en decimal, en binario usando el prefijo % y hexadecimal utilizando el prefijo \$ ejemplo:
 12 es igual que %1100 y también es igual a \$C

<code>Trsb = 0</code>	: indica que todos los pines del puerto B son de salida
<code>inicio:</code>	: nombre de subrutina inicio
<code>portb = %00000001</code>	: aunque puede controlar todo el puerto B, solo enciende el B0
<code>PAUSE 1000</code>	: espera un segundo
<code>portb = %00000000</code>	: obliga a apagarse a todos los pines del puerto B
<code>PAUSE 1000</code>	: espera un segundo
<code>GOTO inicio</code>	: continúa el programa para siempre

Figura 4.4.2. Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo manejando el puerto B, noten que es capaz de manipular todos los pines del puerto B desde el B0 que es el primero de la derecha hasta el B7 el último.

Seguimos con otra forma de programar un parpadeo de un led, esta vez como el ejemplo anterior pero con la diferencia de que solo manejaremos un pin y no todos en conjunto,

<code>Trsb=%0</code>	: indica que solo el puerto B.0 es de salida
<code>inicio:</code>	: nombre de subrutina inicio
<code>portb.0 = 1</code>	: sacar un uno lógico por el puerto B.0
<code>PAUSE 1000</code>	: espera un segundo
<code>Portb.0 = 0</code>	: hacer cero lógico el puerto B.0
<code>PAUSE 1000</code>	: espera un segundo
<code>GOTO inicio</code>	: continúa con la línea de nombre inicio

Figura 4.4.3. Otra forma de programar un parpadeo de un LED a intervalos de 1 segundo manejando el puerto B.0 únicamente como salida.

Como podemos ver hay distintas formas de escribir un programa y todos tienen el mismo resultado final, así que si un proyecto no les funciona bien, prueben escribiendo de una forma diferente.

CAPÍTULO 5

PROYECTOS CON MICROCONTROLADORES PIC

5. PROYECTOS DE APLICACIÓN.

Este Capítulo es el más importante y el más extenso de este libro, los microcontroladores se aprende desarrollando prácticas reales, no hay nada más emocionante y satisfactorio que ver funcionando un proyecto realizado por uno mismo. Al igual que otras carreras la práctica es lo que nos hace mejores, por ejemplo un médico cirujano graduado aprenderá mucho más en la vida real mientras más cirugías realice, una persona que tomó cursos de guitarra aprenderá a tocar cada vez mejor mientras más practique con la guitarra, así mismo nosotros aprenderemos mucho de los PIC'S mientras más proyectos nos proponemos a realizar.

Es importante seguir en orden el avance de los proyectos ya que existen proyectos que requieren de una secuencia de aprendizaje por ejemplo no podremos entender bien como funciona el proyecto 5.2.2.Luces del auto fantástico, si no hicimos el proyecto que explica cómo hacer repeticiones el del literal 5.2.1 Ejercicio con la instrucción FOR NEXT.

Como materiales básicos necesitamos un PIC16F62X, un protoboard preferible de 4 regletas, un regulador de voltaje 7805, una fuente de energía y por supuesto tener un *grabador de PIC'S* como el que se incluye en este libro, este es un grabador tipo JDM (Jens Dyekjaer Madesn), muy fácil de utilizar ya que solo requiere la energía del puerto com. Para construir este grabador será necesario que primero lea el capítulo 6, donde se enseñará paso a paso como ir armando el grabador.

Adicionalmente para ayuda del lector se incluye en el CD todos los ejercicios que se presentan en este capítulo, tanto en extensión .bas y .hex. Además si desea utilizar otro modelo de PIC que no sea PIC16F62X, debe considerar los cambios necesarios para su correcto funcionamiento, por ejemplo para micros que tienen conversores A/D (16F87X,16F81X), se debe reemplazar la línea CMCON=7 por ADCON1=7, (ver literal 5.10.1).

NOTA: Para mayor facilidad en la escritura de los programas se ha cambiado el nombre de los pines de los puertos, por ejemplo: el pin RB6 se lo llamará únicamente B.6 o simplemente B6, el puerto RB5, se lo llamará B.5 o B5, y así con todos los demás puertos. Otro punto importante que ustedes observarán en los proyectos es que no hay diagramas de flujos, esto se debe a que no es necesario, porque el programa escrito en sí es como un diagrama de flujo.

5.1 PROYECTOS CON LEDS

5.1.1 PROGRAMA BÁSICO PARA HACER PARPADEAR UN LED CON INTERVALOS DE 1 SEGUNDO.

Este proyecto ya revisamos muchas veces anteriormente, si usted no ha leído todo el libro lea por lo menos la página 21 y la 22 y de seguro entenderá el siguiente programa, adicionalmente puede abrir el programa que se encuentra listo en el CD en Ejercicios\ led intermitente.bas.

```
led VAR portb.0      ; etiqueta para el puerto B.0
pepe:                ; nombre de subrutina pepe
    HIGH led         ; enciende el led que esta conectado en el pin 6
    PAUSE 1000       ; espera un segundo
    LOW led          ; apaga el led
    PAUSE 1000       ; espera un segundo
GOTO pepe            ; continúa el programa desde pepe para siempre
END                  ; fin de las instrucciones
```

Figura 5.1.1.1. led intermitente.bas Programa para el PIC16F628 que hace parpadear un led.

Una vez escrito el programa compilamos y grabamos el PIC, si no sabe como hacerlo recomendamos leer las páginas 29 a la 32, no olvide poner en el IC-prog oscilador **intRC I/O** y deshabilitar el **MCLR**, luego de que todo este bien conectamos como ilustra la siguiente figura:

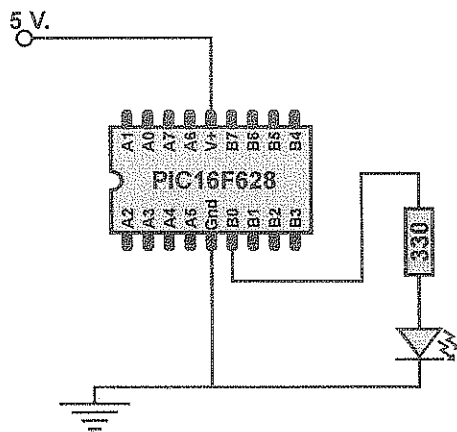


Figura 5.1.1.2. Diagrama de conexión de un led en el puerto B.0 o pin 6 para hacer un parpadeo de un led.

Una vez realizado este proyecto siga intentando con diferentes tiempos de **PAUSE**, recuerde que son en milisegundos y sus valores son desde 1 hasta 65535, pruebe con **PAUSE 100** verá que el parpadeo es más rápido y **PAUSE 2000** es más lento, así mismo ponga de diferentes valores entre los 2 **PAUSES** ejemplo el primer **PAUSE** coloque **PAUSE 2000** y en el segundo coloque **PAUSE 500** verá diferentes efectos.

NOTA: recuerde que el PIC ejecuta cada línea de programa en 1 μs . (0,000001 segundos) por lo que si no pusieramos uno de los 2 **PAUSES** veríamos solo encendido o solo apagado el LED esto se debe a que no hay tiempo para ver el efecto de transición del LED.

Para entender mejor pruebe el siguiente programa en el que se eliminó el segundo **PAUSE 1000**:

led VAR portb.0	; etiqueta para el puerto B.0
pepe:	; nombre de subrutina pepe
HIGH led	; enciende el led que esta conectado en el pin 6
PAUSE 1000	; espera un segundo
LOW led	; apaga el led
GOTO pepe	; continúa el programa desde pepe para siempre
END	; fin de las instrucciones

Figura 5.1.1.3. Programa para entender la velocidad a la que trabaja el PIC.

Si ya colocó en el protoboard verá que el LED permanece solo encendido, pero no es así el LED se apaga, pero por un período muy corto lo que a simple vista no lo notamos pues este dura 3 Us., el tiempo en que el PIC cambia a la siguiente instrucción. Analicemos detenidamente lo que hace el PIC desde el momento en que corre la línea **HIGH led**, en ese mismo instante se enciende el LED luego pasa 1 Us. y ejecuta el **PAUSE 1000**, este es un grupo de subrutinas que el compilador pbp genera para dar un retardo de 1 segundo sin hacer nada, por su puesto el LED sigue encendido porque aún no lo decimos que se apague. Una vez terminado el período del **PAUSE 1000** pasa a **LOW led** en 1 Us., en este mismo instante se apaga el LED, pero la siguiente línea no es otro pause sino ir a pepe y esto se demora 2 Us. por lo que enseguida se enciende el LED al llegar a **HIGH led**.

5.1.2. UN SEMÁFORO DE 2 INTERSECCIONES.

Recuerda que en la página 34 se habla de manejar un grupo de leds, pues bien este es el ejemplo ideal para entender cuando utilizar **HIGH** y cuando **PORT**, yo recomiendo utilizar **HIGH** cuando se trata de un solo led o relé, etc., pero si vamos a utilizar un grupo de leds veremos que es mucho más fácil si manejamos todo el puerto sea este el A o el B, pero de todas formas escribiremos el programa de las dos maneras, y ustedes se darán cuenta cual es la forma más rápida de programar, en la figura 5.1.2.1 se muestra el diagrama de conexión para este proyecto.

MATERIALES.

- además de los materiales básicos, protoboard, regulador 7805 y fuente de voltaje
- 6 LEDS, 2 rojos, 2 amarillos, 2 verdes, todos de 5mm.
- 6 resistencias de 330Ω a ½ vatio, naranja-naranja-café

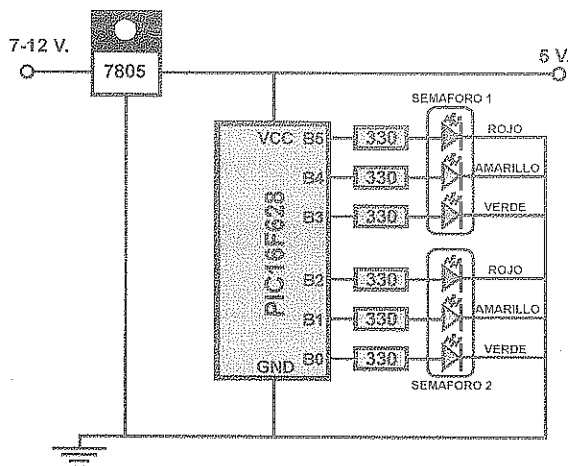


Figura 5.1.2.1 Diagrama esquemático de conexión para un semáforo de 2 intersecciones.

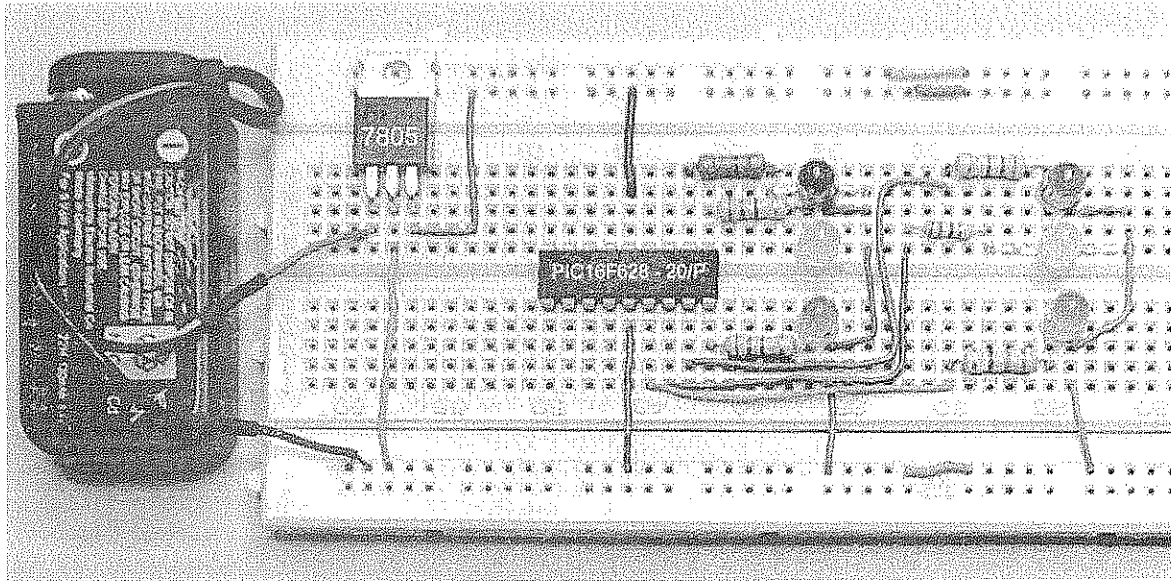


Figura 5.1.2.2. Fotografía del semáforo armado en un protoboard.

Bien es momento de escribir el programa, empecemos de la forma que se maneja el puerto completo, para esto debemos entender como trabaja los semáforos. Primero nunca se encienden las luces del mismo color, es decir no puede estar en el un semáforo verde y en el otro verde también, existe un cambio de verde a amarillo mientras en el otro semáforo sigue en rojo, en el momento que se pone en rojo el primer semáforo el segundo salta de rojo a verde.

Por considerar que esta es una práctica, pondremos tiempos estimados de cambio de color, de verde a amarillo durará 9 segundos, de amarillo a rojo solo 3 segundos.

trisb=0	;indica que todos los pines del puerto B son de salida
semaforo:	;nombre de la línea semáforo
portb=%100001	;encender rojo del 1er semáforo y verde del 2do semáforo
PAUSE 9000	;esperar 9 segundos
portb=%100010	;cambiar en el 2do semáforo de verde a amarillo
PAUSE 3000	;esperar 3 segundos
portb=%001100	;cambiar a verde en el 1er semáforo y rojo el 2do semáforo
PAUSE 9000	;esperar 9 segundos
portb=%010100	;cambiar en el 1er semáforo de verde a amarillo
PAUSE 3000	;esperar 3 segundos
GOTO semaforo	;continuar con el ciclo para siempre
END	; fin de la programación

Figura 5.1.2.3. semaforo.bas Programa del semáforo manejando el puerto B completo.

El siguiente es otra forma de escribir el programa, el semáforo funciona igual.

rojo1	VAR portb5	; etiquetas para los puertos
amarillo1	VAR portb.4	
verde1	VAR portb3	
rojo2	VAR portb2	
amarillo2	VAR portb1	
verde2	VAR portb.0	

continúa.....

```

semáf:
HIGH rojo1 : HIGH verde2           ;primer semáforo en rojo y 2do en verde
    PAUSE 9000                      ;esperar 9 segundos
LOW verde2 : HIGH amarillo2         ;2do semáf. Pasa de verde a amarillo
    PAUSE 3000                      ;esperar 3 segundos
LOW amarillo2 : LOW rojo1 : HIGH verde1 : HIGH rojo2 ;1er semáf. Verde 2do se. rojo
    PAUSE 9000                      ;esperar 9 segundos
LOW verde1 : HIGH amarillo1         ;1er semáforo cambia de verde a amarillo
    PAUSE 3000                      ;esperar 3 segundos
GOTO semáf                          ;continuar el programa desde semáf
END                                 ;fin de la programación

```

Figura 5.1.2.4. Programa del semáforo manejando pin por pin con HIGH y LOW.

En esta otra manera de escribir el programa, noten que es más largo que el primer programa, y además no escribimos `tristb=0` al inicio, porque **HIGH** ya los convierte en salida, también aquí se ve algo nuevo los 2 puntos (:), estos sirven para declaraciones múltiples en una sola línea, para ambos casos el tamaño de código generado es el mismo.

Ejemplo, si queremos expresar en una sola línea las 2 siguientes declaraciones:

```

HIGH rojo1
HIGH verde2

```

quedaría así : **HIGH rojo1 : HIGH verde2**

NOTA: si desea aplicar este proyecto con focos de 110v., debemos utilizar periféricos de salida como los relés, el siguiente es el diagrama de conexionado de un relé:

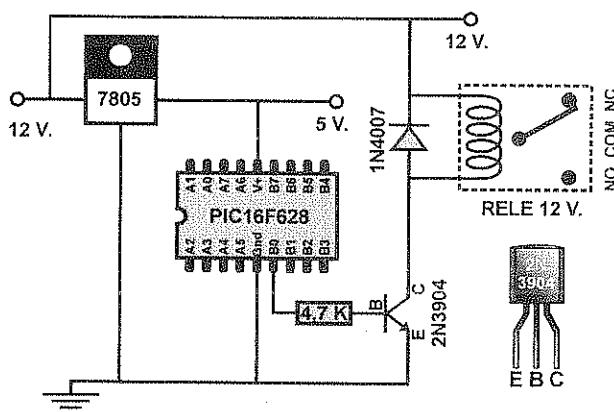


Figura 5.1.2.5. Diagrama esquemático de Conexión de un relé al PIC.

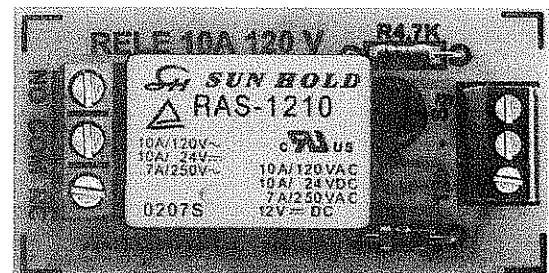


Figura 5.1.2.6. Fotografía de un módulo periférico de salida con un relevo.

5.1.3. JUEGO DE LUCES PARA DISCOTECA.

Este proyecto propone familiarizar aún más con el manejo de los puertos, esta vez vamos a utilizar las 8 salidas del puerto B, se trata de una secuencia de luces que deben encenderse de

izquierda a derecha una tras otra con un intervalo de 200 milisegundos. En la figura 5.1.3.1.se muestra como se debe conectar cada uno de los LEDS.

MATERIALES.

-8 LEDS de 5mm.

-8 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café

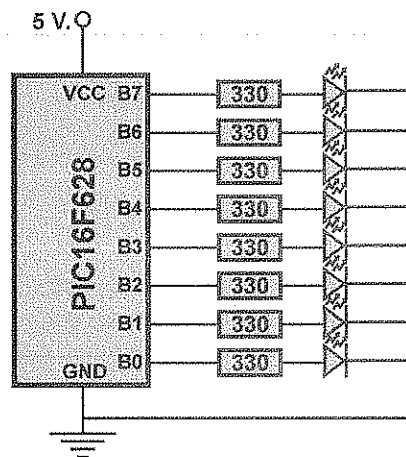


Figura 5.1.3.1. Diagrama esquemático de conexión de 8 LEDS, para el proyecto de luces de discoteca con PIC. Debido a la capacidad de corriente que puede entregar este PIC, no se necesita de buffers amplificadores.

A continuación en el siguiente programa veremos una declaración nueva el **GOSUB** y el **RETURN**, estos sirven para cuando tenemos muchas repeticiones de una línea o grupo de líneas de programa, en nuestro caso el **PAUSE 200**, en vez de poner en cada cambio de estado de las luces, lo agruparemos en una sola subrutina y lo llamaremos las veces que queramos, la declaración **RETURN** lo envía de regreso a continuar después del **GOSUB** que lo envió. Una de las ventajas más importantes que tenemos es que ahorra espacio en el PIC, y otra que si queremos cambiar el **PAUSE 200** por el de otro valor, basta con cambiar una sola vez y el cambio se ejecuta para todos, lo que al contrario si no lo utilizáramos el **GOSUB** y escribiríamos 30 **PAUSE 200**, deberíamos cambiarlo a los 30 **PAUSES** por el de otro valor, un ejemplo sería:

Prueba1:

```
Portb=%00000001
PAUSE 1000
Portb=%00000010
PAUSE 1000
Portb=%00000100
PAUSE 1000
```

GOTO prueba1

Prueba2:

```
Portb=%00000001
GOSUB pablo
Portb=%00000010
GOSUB pablo
Portb=%00000100
GOSUB pablo
GOTO prueba2
```

Pablo:

```
PAUSE 1000
```

RETURN

Figura 5.1.3.2. Si quisiéramos cambiar el PAUSE 1000 por PAUSE 500, en el programa de la izquierda deberíamos cambiar a cada uno de ellos, en total 3, pero para el de la derecha basta con cambiarle al que está dentro de la subrutina pablo, y tendríamos el mismo resultado.

Trisb=%00000000	;convierte todos los pines del puerto B en salidas
discoteca:	;nombre de la subrutina
Portb=%00000001	;enciende el puerto B.0, los demás permanecen apagados
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%00000010	;enciende el puerto B.1, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%00000100	;enciende el puerto B.2, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%00001000	;enciende el puerto B.3, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%00010000	;enciende el puerto B.4, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%00100000	;enciende el puerto B.5, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%01000000	;enciende el puerto B.6, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
Portb=%10000000	;enciende el puerto B.7, los demás les apaga
GOSUB pedro	; ir a subrutina tiempo y volver cuando diga RETURN
GOTO discoteca	; ir al inicio del programa
pedro:	;esta es la subrutina pedro
PAUSE 200	;retardo de 200 milisegundos, aquí podemos cambiarlo
RETURN	;volver al GOSUB que le envió

Figura 5.1.3.3. discoteca.bas Programa para las luces de discoteca.

NOTA: la ubicación de la subrutina pedro, es importante fíjense que se encuentra después y fuera de las líneas principales de programación, si esta misma subrutina lo colocáramos al principio del programa, de seguro se nos cuelga porque al encontrar un RETURN simplemente no sabe a donde retornar ya que nadie lo ha enviado aún.

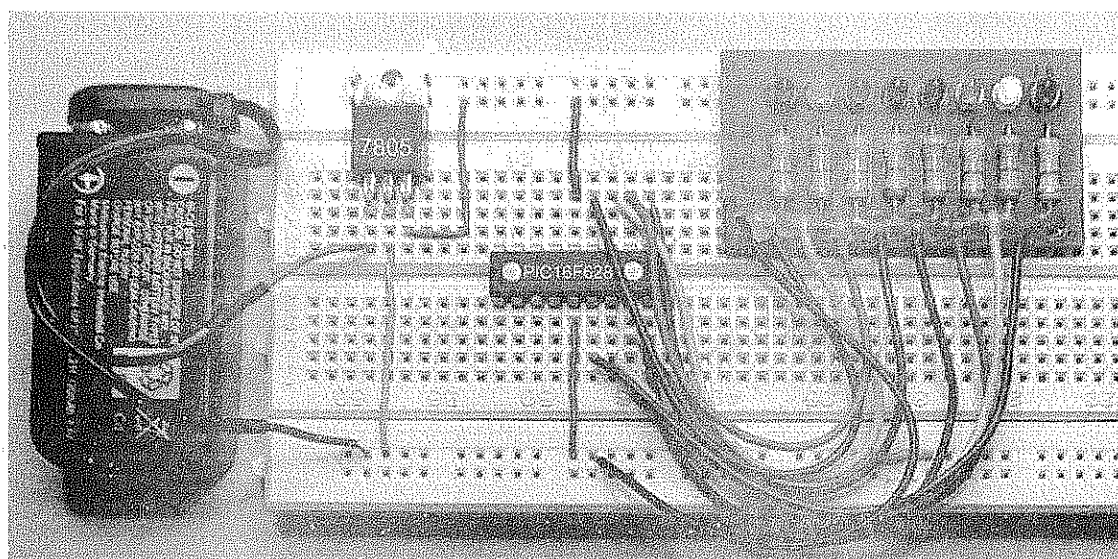


Figura 5.1.3.4. Fotografía del proyecto de luces para discoteca, se muestra un módulo de 8 leds con sus resistencias, basta con conectar el 1er cable a tierra y los demás a cada uno de los pines del puerto B del PIC, esto ahorra tiempo de instalación.

5.2 PROYECTOS DE REPETICIONES

5.2.1. EJERCICIO CON LA INSTRUCCIÓN FOR NEXT.

Este proyecto es muy importante entenderlo, ya que el siguiente proyecto de luces de auto fantástico también utiliza la declaración **FOR NEXT**.

Esta declaración sirve para ejecutar un número *n* de veces una línea de programa o grupo de líneas de programa, el siguiente proyecto pretende encender un led en el puerto B.0 5 veces con intervalos de ½ segundo, después debe detenerse por 2 segundos y luego parpadear 3 veces más, detenerse por 3 segundos y luego repetir nuevamente el proceso, se puede utilizar el proto que se armó para las luces de discoteca ya que el mismo nos servirá después para el siguiente proyecto el de las luces del auto fantástico, **FOR NEXT** se utiliza de la siguiente manera:

```
Peter VAR BYTE      ;primero se crea y asigna un tamaño para la variable peter
FOR peter = 1 TO 5   ;Ejecuta las siguientes instrucciones 5 veces hasta donde dice NEXT
.....              ;una vez concluido las repeticiones continúa con la declaración que está
.....              ;después del NEXT, peter debe ser creado como variable, es decir
NEXT                 ;asignarle un espacio en la memoria en este caso para 5.
```

LAS VARIABLES BIT, BYTE Y WORD. Estas son creadas para guardar datos en la memoria RAM (Random Access Memory) o memoria de acceso casual, esta memoria trabaja únicamente mientras esté alimentado el PIC, una vez que el PIC es desconectado, los datos de la memoria RAM se borran.

Para crear una variable es muy similar a asignar un nombre de un pin, como peter **VAR** portb.3, la diferencia está en que en vez de poner el pin se pone el tamaño de la memoria a utilizar y estos son los siguientes:

```
Peter VAR BIT        ; crea una variable y asigna un tamaño de un bit es decir 0 o 1
Peter VAR BYTE        ; crea una variable y asigna un tamaño de 8 bits es decir de 0 a 255
Peter VAR WORD        ; crea una variable y asigna un tamaño de 2 bytes es decir de 0 a 65535
```

Para nuestro caso como queremos hacer 5 repeticiones, nos corresponde crear un **BYTE** que nos permite almacenar un número hasta el 255.

```
repe VAR BYTE        ;crea la variable repe y le asigna un espacio de memoria de 0 a 255
Led1 VAR portb.0      ;asigna el nombre de led1 al pin B.0
programa:              ;nombre de la línea programa
FOR repe = 1 TO 5      ;para repeticiones de 1 a 5 veces
    HIGH led1          ; encender el LED
    PAUSE 500           ; esperar 0,5 segundos
    LOW led1           ; apagar el LED
    PAUSE 500          ; esperar 0,5 segundos
    NEXT               ; siguiente repetición hasta que sea repe = 5
    PAUSE 2000         ; esperar 2 segundos

FOR repe = 1 TO 3      ;para repeticiones de 1 a 3 veces
    HIGH led1          ; encender el LED
```

Continúa

PAUSE 500	; esperar 0,5 segundos
LOW led1	; apagar el LED
PAUSE 500	; esperar 0,5 segundos
NEXT	; siguiente repetición hasta que sea repe = 3
PAUSE 3000	; esperar 3 segundos
GOTO programa	; ir a programa
END	; fin de la programación

Figura 5.2.1.1. repeticiones.bas Programa para encender un led n número de veces.

5.2.2. LUCES DEL AUTO FANTÁSTICO (DESPLAZAMIENTOS).

Este proyecto es muy similar al de las luces para discoteca, con la diferencia de que este se enciende de izquierda a derecha y luego de derecha a izquierda, pensaríamos que el programa sería el doble del tamaño que el que hicimos para las luces de discoteca, pero no es así, recuerden que existen varios caminos para llegar a un mismo objetivo, y este es uno de ellos, esta vez utilizaremos los desplazamientos, que no son nada más que recorrer un uno lógico a la izquierda o a la derecha de la salida de los puertos.

Los desplazamientos utiliza la multiplicación y la división, como sabemos el PIC trabaja con el sistema binario, si tenemos una variable X con un valor inicial de 1 (%00000001) y lo multiplicáramos por 2, el resultado sería 2 (%00000010), y este a su vez lo volveríamos a multiplicar por 2 el resultado sería 4 (%00000100), y así sucesivamente hasta llegar a 128, en donde en binario sería (%10000000), veríamos que los leds se enciende de la misma forma que en las luces para discoteca, para hacer que las luces se regresen hasta el puerto B.0 debemos dividir para 2, entonces $128 / 2$ es igual a 64 (%01000000), como podemos ver ahora está regresando a su lugar de origen, los desplazamientos se escribe de la siguiente manera:

LEDS = LEDS << 1	equivale a multiplicar por 2 y se desplaza uno a uno, también podemos
LEDS = LEDS << 2	equivale a multiplicar por 4 y se desplaza de dos en dos
LEDS = LEDS >> 1	equivale a dividir por 2 y se desplaza uno a uno hacia la derecha.

Entendido como funciona los desplazamientos desarrollamos el siguiente programa:

x VAR BYTE	; creamos la variable x y le asignamos tamaño de 255
LEDS VAR PORTB	; todo el puerto B se llamará LEDS
TRISB = 0	; hacemos salidas a todo el puerto B
 LEDS = 1	 ; Cargamos la variable LEDS con 1 (%00000001)
PROG:	
FOR x = 1 TO 7	; repetir 7 veces
LEDS = LEDS << 1	; desplazar uno a uno a la izquierda
PAUSE 200	; esperar 200 milisegundos
NEXT	; repetir hasta que x sea = a 7
 FOR x = 1 TO 7	 ; repetir 7 veces
LEDS = LEDS >> 1	; desplazar uno a uno a la derecha
PAUSE 200	; esperar 200 milisegundos
	continúa ...

```

NEXT          ;repetir hasta que x sea = a 7
GOTO PROG    ; ir a PROG
END          ; fin de la programación

```

Figura 5.2.2.1. auto fantastico.bas Programa para encender las luces del auto fantástico.

NOTA: No olviden cargar a $LEDS = 1$, porque si no lo hace significa que vale cero (0) y esto multiplicado por 2 siempre dará cero, en consecuencia nunca veremos el desplazamiento.

5.3 PRÁCTICAS CON PULSADORES

5.3.1. EJERCICIO CON PULSADORES.

Para estas prácticas debemos entender como funcionan los pulsadores con el PIC, existen básicamente 2 tipos de conexión para los pulsadores, el que siempre está en 1 lógico (5 V.) y cuando es pulsado cambia a cero lógico (0 V.), y el que está en cero lógico y cuando pulsamos pasa a uno lógico, los siguientes son los diagramas de conexión:

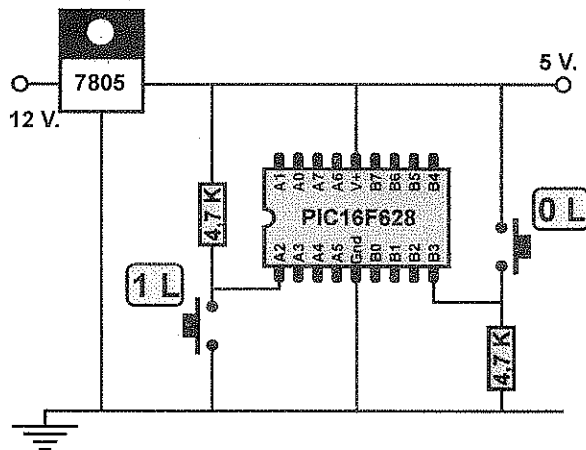


Figura 5.3.1.1. Diagrama de conexión de 2 pulsadores, el primero es 1 Lógico, es decir siempre permite el ingreso de 5 V. al PIC, cuando es presionado, el voltaje se desvía a tierra y en ese caso el PIC detecta un cambio de estado de 1 L a 0 L el funcionamiento del segundo pulsador es totalmente lo contrario.

Existe otra manera de hacer un pulsador o entrada con más voltaje del que el PIC soporta, y es haciendo un divisor de voltaje, esto es muy utilizado para indicar si una batería de 12 voltios por ejemplo, se encuentra cargada o descargada.

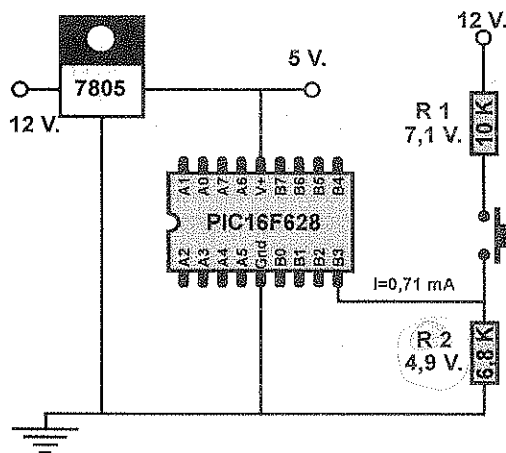


Figura 5.3.1.2. Diagrama de conexión de un pulsador 0 lógico a una fuente de 12 V. si se le conecta como en el de la figura anterior es muy probable que el PIC se dañe, por esta razón se le conecta en el divisor de voltaje, en donde el voltaje baja a 4,9V. y la corriente que el PIC debe soportar es de 0,71 mA.

Ejercicio:

Calcular la resistencia R_2 , para hacer un divisor de voltaje en el que salga aproximadamente 5 Voltios, si la fuente es una batería de 24 Voltios DC.

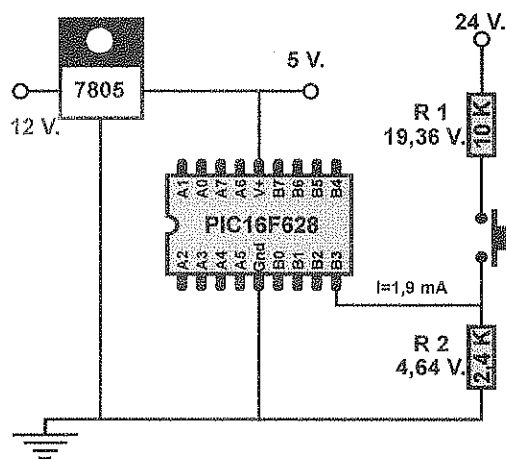


Figura 5.3.1.3. Diagrama de un pulsador para una fuente de 24 V.

Explicación.-necesitamos un divisor de voltaje de 5 V., por lo que en R1 debería caer 19 V.

$$V1 = \frac{Vt \times R1}{Req} \quad Req = \frac{Vt \times R1}{V1} \quad Req = \frac{24 \times 10000}{19}$$

$$Req = 12631,6\Omega \quad R2 = R1 - Req \quad R2 = 2631,6\Omega$$

Tenemos en el mercado de 2,7 K y 2,4 K, no podemos usar la de 2,7K porque el voltaje pasaría de 5V. utilizaremos la de 2,4 K y tendremos los siguientes cálculos:

$$V1 = \frac{Vt \times R1}{Req} \quad V1 = \frac{24V \times 10000\Omega}{12400\Omega}$$

$$V1 = 19,355 V.$$

$$V2 = \frac{Vt \times R2}{Req} \quad V2 = \frac{24V \times 2400\Omega}{12400\Omega}$$

$$V2 = 4,645 V.$$

La corriente que circula por el PIC sería:

$$I = \frac{Vt}{Req} \quad I = \frac{24}{12400} \quad I = 1,93 \text{ mA.}$$

;Lo cual estaría bien, considerando que el PIC soporta 25 mA en modo sumidero por cada pin.

LA DECLARACIÓN IF ... THEN. Esta sirve de condicionante, si es verdadera ejecuta la operación que sigue al **THEN**, y si es falsa salta a la siguiente línea después del **THEN**, existen varias formas de aplicación:

IF portb.0 = 0 **THEN** Pablo ; ir a Pablo si la entrada portb.0 es cero lógico

IF portb.6 = 1 **THEN** juan ; ir a juan si la entrada portb.6 es 1 lógica

IF portb.0 = 0 **THEN** ; si la comparación es verdadera ejecuta todo el contenido que
HIGH porta.2 ; se encuentra entre el **THEN** y el **ENDIF**
PAUSE 1000
LOW porta.2
ENDIF

IF portb3 = 0 **THEN** ; si la comparación es verdadera ejecuta el contenido entre
..... ; **THEN** y **ELSE** y si es falsa ejecuta el contenido entre **ELSE**
ELSE ; y **ENDIF**
.....
ENDIF

IF repe > 35 **THEN** iniciar ;si la variable rep es mayor que 35 ejecuta iniciar, además se
;puede utilizar los demás operadores: =, != (NO ES IGUAL), <, >, <=, >=

IF (porta.1=0) **AND** (porta.2=0) **THEN** prog ;si porta.1 y porta.2 son igual a cero ejecuta prog,
así mismo soporta los demás operadores como: **OR**, **XOR**, **NOT AND**, **NOT OR**, **NOT XOR**

El siguiente es el diagrama de conexión para esta práctica de pulsador.

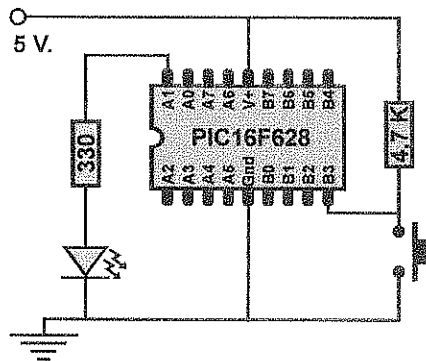


Figura 5.3.1.4. Diagrama de conexión de un pulsador conectado en el puerto B.3, de estado 1 lógico, en el momento que es presionado este desvía la corriente hacia tierra, por lo que el PIC detecta un cambio de estado a cero lógico, en ese instante se enciende el led.

MATERIALES.

- 1 LED de 5mm.
- 1 resistencia de 330Ω a ½ vatio, naranja-naranja-café
- 1 resistencia de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
- 1 pulsador para protoboard normalmente abierto como el de la figura 5.3.3.2

A continuación el programa en BASIC para leer el estado de un pulsador

```

cmcon = 7 ;convierte todo el puerto A en Digital
pro:
  IF portb.3=0 THEN encen ;pregunta si portb.3=0 para ir a encen
  GOTO pro ;ir a pro
encen:
  HIGH porta.1 ;encender el led
  PAUSE 1000 ;esperar 1 segundo
  LOW porta.1 ;apagar el led
  GOTO pro ;ir a pro
END ;fin de la programación
  
```

Figura 5.3.1.5. pulsador.bas Programa para leer un pulsador, si este es presionado se enciende un led que está conectado en el puerto A.1 y se apaga después de 1 segundo.

5.3.2. CONTADOR BINARIO CON PULSADOR ANTIRREBOTE.

En esta práctica haremos un contador binario, el resultado lo veremos en código binario a través de 8 leds conectados en el puerto B. Para esta práctica necesitamos poner un antirrebote al

pulsador, ya que si presionamos por un instante, dada la velocidad que procesa el PIC el programa se ejecutaría varias veces hasta que soltemos el pulsador, para aprender de los errores haremos el programa sin antirrebote de tecla y luego con antirrebote.

MATERIALES.

- 8 LEDS de 5mm.
- 8 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 1 resistencia de $4,7 K\Omega$ a $\frac{1}{2}$ vatio, amarillo-violeta-rojo
- 1 pulsador para protoboard normalmente abierto como el de la figura 5.3.3.2

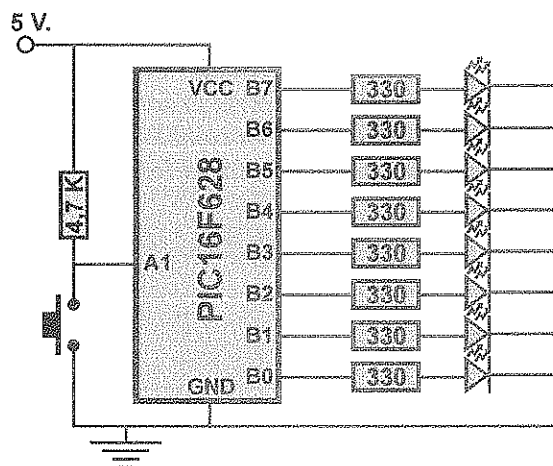


Figura 5.3.2.1. Diagrama de conexión de 8 leds y un pulsador, cada que pulsamos el botón, los leds aumentan en código binario.

```

cmcon = 7           ;convierte todo el puerto A en Digital
Trisb=0             ; hace todos los pines del puerto B como salidas

boton VAR portA.1    ;asigna el nombre de botón al puerto A.1
num VAR BYTE         ;crea la variable num con un tamaño de 255
num= 0               ;carga el valor inicial de 0 a la variable num

pulsar:
    portb=num         ;indica sacar el valor de num a través del puerto B
    IF boton=0 THEN contar ;pregunta si el botón a sido presionado
    GOTO pulsar       ;ir a pulsar , mantiene encerrado

contar:
    num=num + 1       ;suma 1 a la variable num y el nuevo valor le guarda en num
    GOTO pulsar       ; volver al principio del programa

END                  ; fin de la programación
    
```

Figura 5.3.2.2. Programa para sumar en código binario sin antirrebote de tecla.

Si tuvo la oportunidad de ver funcionando este proyecto, se dará cuenta que cada que pulsamos el botón el contador aumenta demasiado, esto es como dijimos antes porque el PIC trabaja a 1Us. cada instrucción y cuando una persona presiona el botón, por lo menos necesita de 100 mS de tiempo para soltarlo, en ese tiempo el PIC ya sumó alrededor de 25.000 veces. Para solucionar este problema proponemos hacer un programa de antirrebote de tecla, en el cual si presionamos el botón, este le envía a un programa que lo mantiene encerrado, y únicamente sale de esta subrutina

en el momento que el pulsador deja de ser presionado, a continuación incluimos un **PAUSE 200**, que es necesario para que en el momento de soltar la tecla, se estabilice la señal.

```

cmcon = 7           ;convierte todo el puerto A en Digital
Trisb=0             ;hace todos los pines del puerto B como salidas

boton VAR portA.1   ;asigna el nombre de botón al puerto A.1
num VAR BYTE        ;crea la variable num con un tamaño de 255
num= 0              ;carga el valor inicial de 0 a la variable num
pulsar:
    portb=num        ;indica sacar el valor de num a través del puerto B
    IF boton=0 THEN contar ;pregunta si el botón a sido presionado
    GOTO pulsar      ;ir a pulsar, mantiene encerrado
contar:
    IF boton=0 THEN contar ;espera a que suelte el boton para continuar
    PAUSE 200         ;espera de 200 mls para estabilizar el botón
    num=num + 1       ;suma 1 a la variable num y el nuevo valor le guarda en num
    GOTO pulsar       ;volver al principio del programa

END                 ; fin de la programación

```

Figura 5.3.2.3. contador binario.bas Programa para sumar en código binario con antirrebote.

Para conocer más sobre los operadores matemáticos disponibles como: resta, multiplicación, división, etc., existentes en el compilador PICBasic PRO, recomendamos ver la ayuda de microcode en Help Topics\PicBasic Pro Basics\Math operators\introduction o visite las webs WWW.frino.com.ar , WWW.todopic.com.ar y descargue un manual de pbp en español.

5.3.3. LED INTERMITENTE DE VELOCIDAD VARIABLE.

Este proyecto se basa en dos pulsadores, el 1ro para aumentar la frecuencia del parpadeo del LED, y el 2do para disminuir la frecuencia de parpadeo. Para esto utilizaremos 2 operadores matemáticos, la suma y la resta, la suma incrementará las repeticiones de una instrucción **FOR NEXT**, que contiene un **PAUSE 5**, mientras que la resta disminuirá las repeticiones del mismo pause. Debemos tener en cuenta que una variable **BYTE** no puede exceder su contenido a más de 255, ni tampoco pasar a valores negativos al ser restado consecutivamente, si excediera el valor de la variable a más de 255, el mismo se carga con valor de cero, y viceversa si el resultado de la resta pasara a -5, la variable se carga con 255.

Para comprobar lo dicho anteriormente, después de hacer la práctica principal, en una práctica aparte eliminen o conviértalos en comentarios las líneas que dice:

```

; IF veces<10 THEN RETURN
; IF veces>150 THEN RETURN

```

La primera se encarga de no permitir que siga restando, para que el tiempo mínimo de pause sea de 50 milisegundos (10 x **PAUSE 5**), mientras que la segunda se encarga de no permitir que siga sumando si el valor de la variable es mayor que 150, este ejecuta un **RETURN**, con esto el tiempo máximo de pause sería de 750 milisegundos (150 x **PAUSE 5**)

MATERIALES.

- 1 LED de 5mm.
- 1 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 2 resistencia de $4,7\text{ K}\Omega$ a $\frac{1}{2}$ vatio, amarillo-violeta-rojo
- 2 pulsadores para protoboard normalmente abierto como los de la figura 5.3.3.2

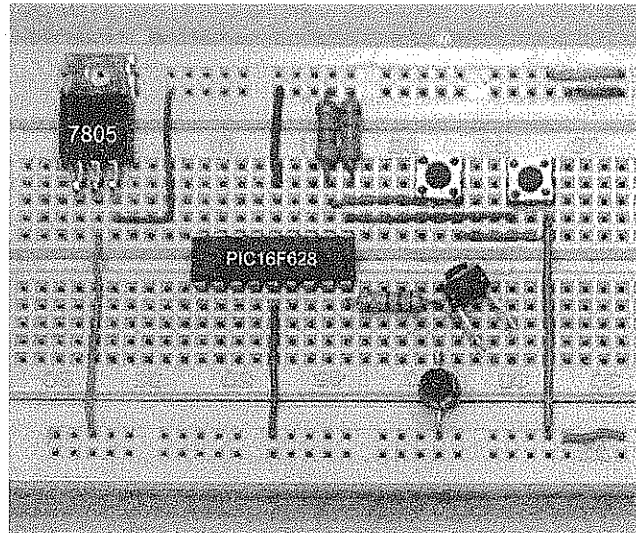
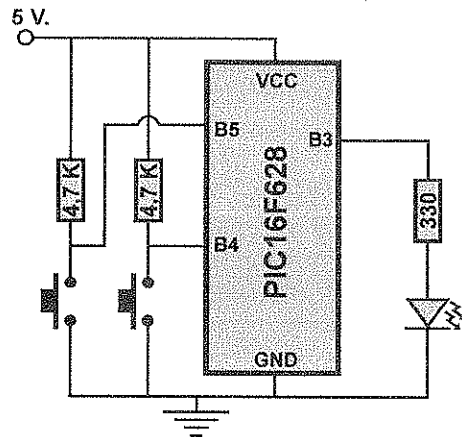


Figura 5.3.3.1. Diagrama de conexión de 2 pulsadores y un led en el puerto B.

Figura 5.3.3.2. Fotografía del diseño armado en el protoboard, también muestra el tipo de pulsador de 2 patitas ideal para los protoboards.

```
Pbaja VAR portb.5 ;el portb.5 se llamará pbaja
Psube VAR portb.4 ;el portb.4 se llamará psube
led VAR portb.3 ;el portb.3 se llamará led
xy VAR byte ;crea la variable xy con tamaño de 255
veces VAR byte ;crea la variable veces con tamaño de 255
veces = 100 ;carga con 100 a la variable veces
```

inicio:

```
HIGH led ;encender el led
GOSUB timer ;ir y retornar de timer
LOW led ;apaga el led
GOSUB timer ;ir y retornar de timer
GOTO inicio
```

timer:

```
IF psube = 0 THEN GOSUB restar ;pregunta si presionó psube
IF pbaja = 0 THEN GOSUB sumar ;pregunta si presionó pbaja
FOR xy = 1 TO veces ;repite desde 1 hasta el valor que contenga veces
pause 5 ;retardo de 5 mls
NEXT ;siguiente repetición
RETURN ;retornar al que le envió
```

Continúa

```

sumar:
IF veces>150 THEN RETURN           ;retorna si veces excede de 150
    veces=veces+5                       ;suma 5 a la variable veces
    RETURN                             ;retorna hacia el que le envió

restar:
    IF veces<10 THEN RETURN          ;retorna si veces es menor que 10
    veces =veces-5                      ;resta 5 a la variable veces
    RETURN                             ;retorna hacia el que le envió

sumar:
IF veces>150 THEN RETURN           ;suma 5 a la variable veces
    veces=veces+5                       ;retorna hacia el que le envió
    RETURN

END                                   ; fin de la programación

```

Figura 5.3.3.3. *led variable.bas Programa para el led intermitente de velocidad variable.*

5.3.4. UTILIZANDO EL MCLR (reset externo).

Este es un reset externo que el PIC posee, aparte del reset al encendido que dispone, la utilización del **MCLR** es muy sencillo, solo debemos instalar un pulsador 1 lógico (pull_up) en el puerto A.5, cuyo pin es específico para el MCLR, el proyecto debe funcionar de la siguiente manera: hacemos un programa para que parpadee un led cada 200 milisegundos (PAUSE 200), para siempre, y al pulsar el botón del MCLR, este parpadeo debe detenerse y al soltarlo debe continuar con el parpadeo del led, es importante tener habilitado el MCLR en el momento de grabar el PIC en el programa IC-prog, el siguiente gráfico muestra como debería estar la configuración del ICprog antes de grabar el PIC:

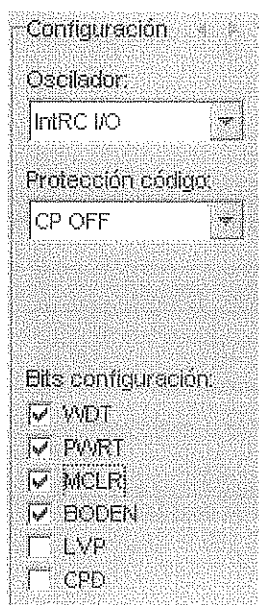


Figura 5.3.4.1. *Configuración del IC-prog, con oscilador Interno RC y habilitado el MCLR (reset externo) necesario para esta práctica.*

MATERIALES.

- 1 LED de 5mm.
- 1 resistencia de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 1 resistencia de $4,7\text{ K}\Omega$ a $\frac{1}{2}$ vatio, amarillo-violeta-rojo
- 1 pulsador para protoboard normalmente abierto

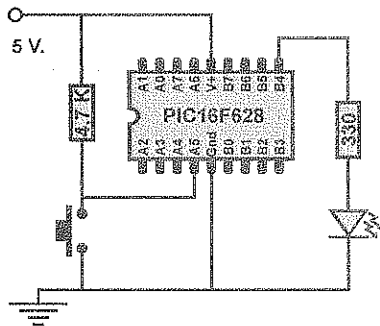


Figura 5.3.4.2. Esquema de conexión de un LED y un pulsador al MCLR (puerto A.5).

iniciar:	:	nombre de subrutina iniciar
HIGH portb.4	:	enciende el led que esta conectado en el pin 10
PAUSE 200	:	espera 200 milisegundos
LOW portb.4	:	apaga el led
PAUSE 200	:	espera 200 milisegundos
GOTO iniciar	:	continúa el programa desde pepe para siempre
END	:	fin de las instrucciones

Figura 5.3.4.3. Programa para un parpadeo de un led cada 200 milisegundos.

Con todo lo aprendido hasta aquí usted podría tranquilamente hacer un PLC como el de la siguiente figura:

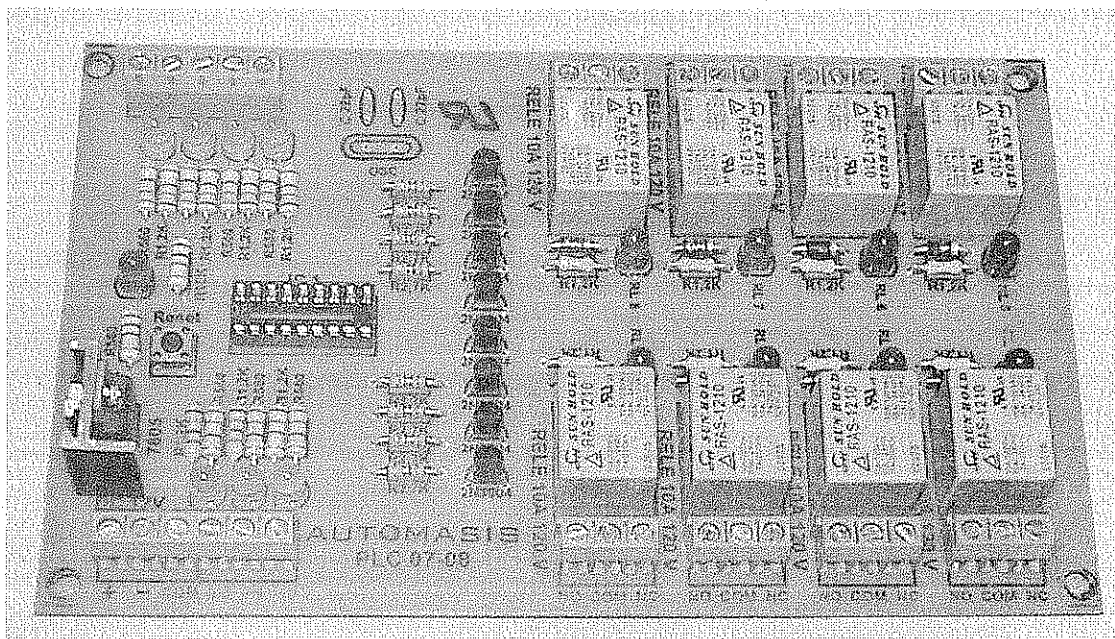


Figura 5.3.4.4. Fotografía de un PLC de 7 entradas y 8 salidas basado en un PIC16F628.

5.4 PROYECTOS CON DISPLAYS

5.4.1. MANEJO DE UN DISPLAY DE 7 SEGMENTOS CON EL CI. 7447.

Los displays son muy utilizados para visualizar datos. Para esta práctica utilizaremos como periférico un display tipo ánodo común, para lo cual facilitamos el diagrama en la figura 5.4.1.2. El proyecto consiste en hacer un contador decimal (0,...,9), con intervalos de 0,5 segundos.

El programa es muy similar al del 5.3.2. contador binario, con la diferencia que solo necesitamos los 4 bits más bajos (B.0, B.1, B.2, y B.3), el decodificador binario a 7 segmentos (7447), es el encargado de transformar el número binario que ingresa a número decimal.

MATERIALES.

- 1 DISPLAY ánodo común preferible como el de la figura 5.4.1.2, ideal para protoboards.
- 7 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 1 CI. 7447 decodificador BCD

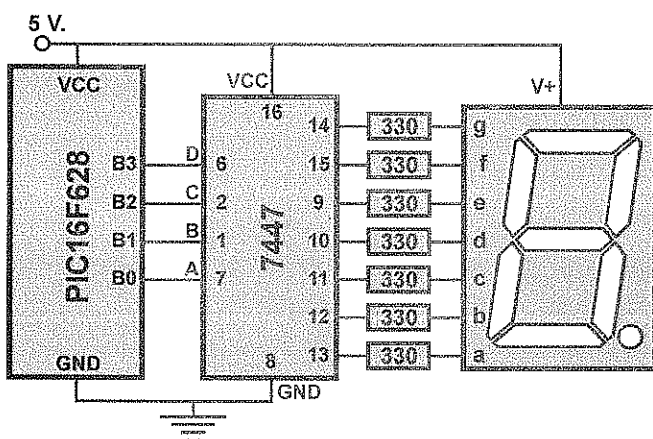


Figura 5.4.1.1. Diagrama de conexión de un display ánodo común con el BCD 7447

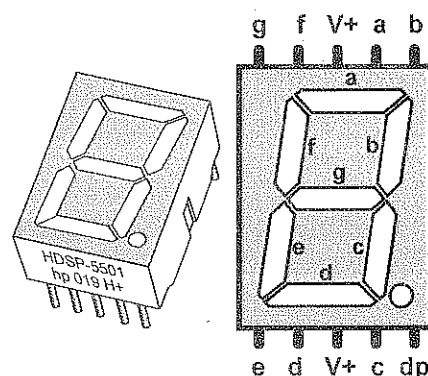


Figura 5.4.1.2. Esquema de pines de un display a. c. HDSP-5501

```

trisb=%11110000      ;hace salidas solo los bits mas bajos de Puerto B
numero VAR BYTE       ;crea la variable número con valor 255

encerar:
    numero = 0         ;carga con cero a la variable numero
display:
    portb=numero       ;sacar por el puerto b el contenido de número
    PAUSE 500          ;esperar 0,5 segundos
    IF numero=9 THEN encerar ;si número es =9 encerar número =0
    numero=numero + 1  ;sumar 1 a la variable número
    GOTO display       ;ir a display
END
    
```

Figura 5.4.1.3. display7seg.bas Programa para el display de 7 segmentos.

5.4.2. UN CONTADOR DECIMAL DE UN DÍGITO CON EL CI. 7447 Y UN PULSADOR.

Adicionando al proyecto anterior un pulsador hacemos un contador manual de un dígito. No debemos olvidar hacer un programa antirrebote de tecla.

MATERIALES.

- 1 DISPLAY ánodo común. preferible como el de la figura 5.4.1.2
- 7 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 1 CI. 7447 decodificador BCD
- 1 pulsador para protoboard normalmente abierto como los de la figura 5.3.3.2

El diagrama de este proyecto es el mismo de la figura 5.4.1.1. pero adicionado un pulsador de estado uno lógico normal en el puerto B.4.

```
trish=%11110000      ;hace salidas solo los bits más bajos de Puerto B
numero VAR BYTE       ;crea la variable número con valor 255
bot VAR portb.4        ;nombre para el puerto B.4
encerrar:
    numero = 0         ;carga con cero a la variable número
display:
    portb=numero       ;sacar por el puerto b el contenido de número
    IF bot=0 THEN aumentar ; si el botón es pulsado ir a aumentar
    GOTO display       ;ir a display

aumentar:
    IF bot=0 THEN aumentar ; si el botón sigue pulsado encerrar en aumentar
    PAUSE 200           ;esperar 0,2 segundos
    IF numero=9 THEN encerrar ;si número es =9 encerrar número =0
    numero=numero + 1   ;sumar 1 a la variable número
    GOTO display       ;ir a display
END
```

Figura 5.4.2.1. display7seg boton.bas Programa para el display de 7 segmentos con pulsador.

Ustedes se preguntarán como hacer para que el número se incremente apenas pulsamos la tecla y no cuando soltamos como actualmente sucede, pues bien para esto utilizamos banderas que no son nada más que una variable de 1 bit, esta nos indica cuando ha sido pulsada. El siguiente es un ejercicio adicional aplicando la bandera.

```
trish=%11110000      ;hace salidas solo los bits más bajos de Puerto B
numero VAR BYTE       ;crea la variable número con valor 255
bot VAR portb.4        ;nombre para el puerto B.4
flag VAR BIT          ;creamos la variable flag de un bit
encerrar:
    numero = 0         ;carga con cero a la variable número
display:
    portb=numero       ;sacar por el puerto b el contenido de número
    IF bot=0 THEN aumentar ; si el botón es pulsado ir a aumentar    continúa .....
```

PAUSE 80	; pause para estabilizar el rebote de la tecla
Flag=0	; cargar la variable con cero
GOTO display	; ir a display
aumentar:	
IF flag = 1 THEN GOTO display	; pregunta si la variable es uno
flag = 1	; cargar la variable con uno
IF numero=9 THEN encerrar	; si número es =9 encerrar número =0
numero=numero + 1	; sumar 1 a la variable número
GOTO display	; ir a display
END	

Figura 5.4.2.2. display7seg boton2.bas Programa para el display de 7 segmentos con pulsador y utilizando una bandera que nos indica si se ha presionado la tecla.

5.4.3. MANEJO DE UN DISPLAY DE 7 SEGMENTOS SIN EL CI. 7447.

Como se dijo en un comienzo el PIC se le puede programar para reemplazar a casi cualquier circuito integrado, en esta ocasión haremos que el propio PIC sea como un CI. 7447, para esto debemos saber que para sacar el número 3 por ejemplo, debemos calcular el número decimal que hace que se enciendan los segmentos correctos del display, esto se hace de la siguiente forma.

Como debemos encender los segmentos a, b, c, d, y g., revisamos los pines del PIC que les corresponde y estos son: B0, B1, B2, B3, Y B6, respectivamente, estos debemos ponerlos en estado cero lógico para que los segmentos se enciendan (recuerde que el display es ánodo común) , y los demás 1 lógico para que permanezcan apagados :

B6	B5	B4	B3	B2	B1	B0
0	1	1	0	0	0	0

, equivale al decimal 48, el display saca el número 3

LA DECLARACIÓN LOOKUP. Sirve para obtener un valor constante de una tabla, esto lo hace según el número de veces que repita el **FOR NEXT**, por ejemplo: la 1ra vez toma el dato que se encuentra en el lugar 0, es decir el Nro. 64, la segunda el Nro.121, y así sucesivamente, y lo va guardando en la variable dat.

MATERIALES.

- 1 DISPLAY ánodo común. preferible como el de la figura 5.4.1.2
- 7 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café

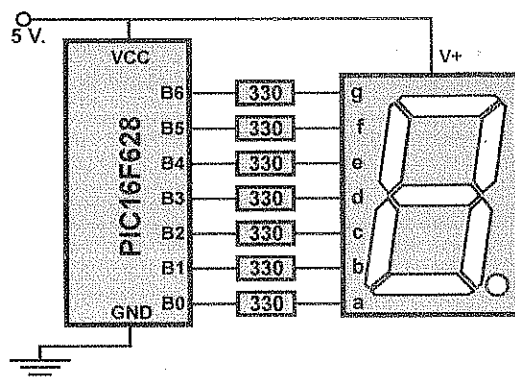


Figura 5.4.3.1. Diagrama de conexión de un display de 7 segmentos directamente al PIC.

```

di VAR BYTE ;crea variable di
dat VAR BYTE ;crea variable dat
TRISB = 0 ;todo el puerto B como salida
prog:
FOR di=0 TO 15 ;para repeticiones de 0 a 15
LOOKUP di,[64,121,36,48,25,18,2,120,0,16,8,3,70,33,6,14],dat ;toma uno por uno cada
; valor de la tabla constante y lo guarda en la variable dat
portb=dat ;sacar el contenido de dat por el puerto B
PAUSE 500 ;espera de 0,5 seg.
NEXT di ;siguiente repetición
GOTO prog
END

```

Figura 5.4.3.2. display7seg lookup.bas Programa para manejar el display directamente.

Como práctica le proponemos que presente por el display la palabra PACO, calcular los números correspondientes y también cuantas repeticiones debe poner en la instrucción FOR NEXT.

NOTA: En esta práctica podemos utilizar un pulsador en el MCLR el del ejercicio 5.3.4., con la finalidad de poder resetear el programa y así volver a empezar a contar nuevamente desde 0.

5.4.4. MANEJO DE 4 DISPLAYS DE 7 SEGMENTOS CON EL CI. 7447.

El siguiente proyecto debe encender 4 displays para poder mostrar cualquier número desde el 0 hasta el 9999, esto lo conseguimos gracias al transistor tipo PNP, que nos ayudará a multiplexar cada uno de los displays, el funcionamiento es bastante sencillo, vamos a conectar los 4 bits más altos a cada transistor y los cuatro bits más bajos al CI. 7447, si por ejemplo queremos sacar el número 6874, primero habilitamos el 4to transistor, el de la derecha y enviamos el número 4, el CI. 7447 se encarga de formar el 4 en el display, luego pasamos a cero lógico el 2do transistor, y los demás lo mantenemos en nivel alto, al mismo tiempo sacamos el número 7 por los bits menos significativos del puerto B, y así consecutivamente, el tiempo que debemos mantener activado cada transistor no puede ser mayor que 5 milisegundos, es decir que los cambios son tan rápidos que el ojo humano ve todos los displays encendidos al mismo tiempo, cuando en realidad solo se enciende uno a la vez.

Ejemplo: para sacar el Nro 8 en las centenas debemos sacar (176+8), es decir el número 184 porque si analizamos en código binario, tenemos que los bits más bajos entran al CI. 7447, y los bits más altos, son los encargados de encender el display que le corresponde a las centenas.

184 = % 10111000

1011 1000

Este número entra al 7447 el cual saca el 8
Este habilita el 3er transistor, el de las centenas

MATERIALES.

- 4 DISPLAYS ánodo común
- 4 transistores 2N3906
- 7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
- 4 resistencias de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo
- 1 CI. 7447 decodificador BCD.


```

trisb=0           ; convierte en salida todo el puerto B
display:
  portb=224+8     ;%11100000, activa el transistor de las unidades y presenta el 8
  PAUSE 5
  portb=208+7     ;%11010000, activa el transistor de las decenas y presenta el 7
  PAUSE 5
  portb=176+6     ;%10110000, activa el transistor de las centenas y presenta el 6
  PAUSE 5
  portb=112+5     ;%01110000, activa el transistor de los miles y presenta el 5
  PAUSE 5
  GOTO display    ; encierra en este lazo
END

```

Figura 5.4.4.3. numero 5678.bas Programa para manejar 4 displays.

Como experimento para comprobar que el PIC solo está activando un display a la vez, cambie todos los pauses a PAUSE 150.

5.4.5. CONTADOR DECIMAL DE 4 DÍGITOS CON EL CI. 7447.

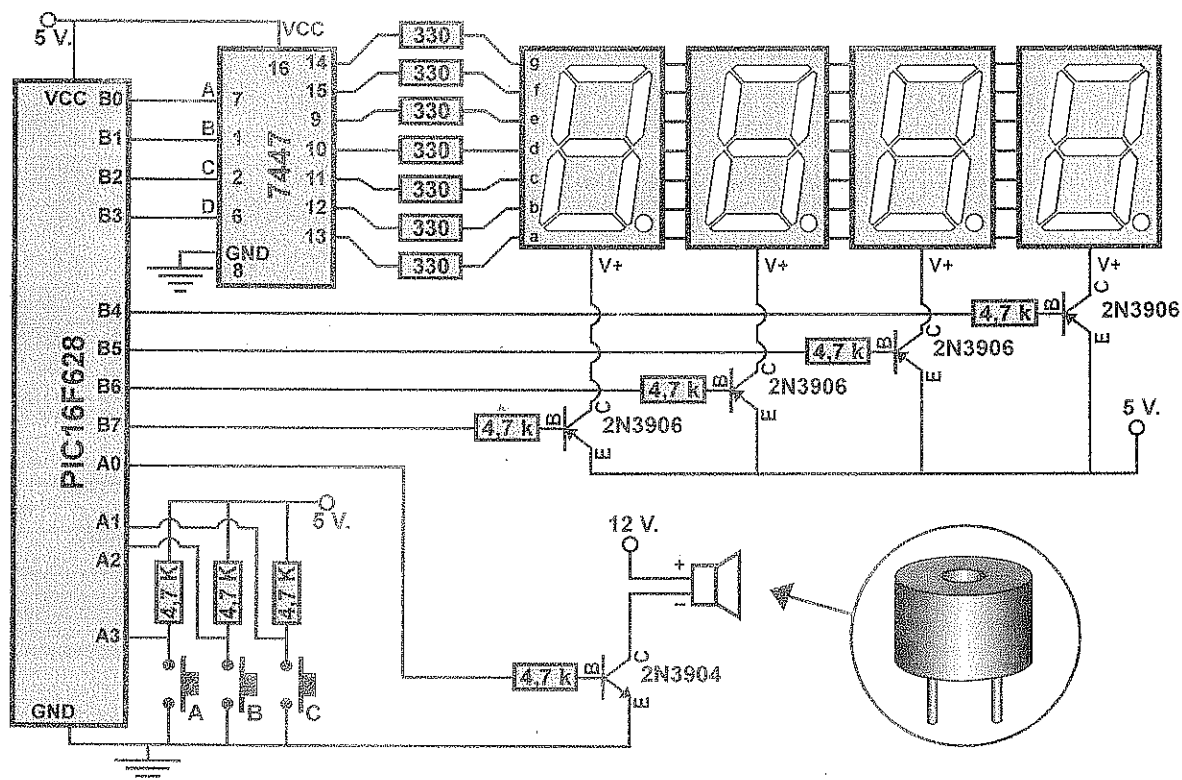


Figura 5.4.5.1. Esquema de conexionado del contador de 4 dígitos, adicionalmente se muestra la forma de la chicharra a 12V de referencia GS1212, esta tiene 2 alambres rojo y negro.

Es hora de hacer un proyecto de un considerable tamaño, una vez entendido como multiplexar 4 displays, y entendido el ejercicio del contador con una bandera de activado el del 5.4.2.2. pues el siguiente proyecto consta en hacer un contador decimal que incremente su valor cada vez que se pulsa el botón A, si pulsamos el botón B se encera y apaga la chicharra, y si pulsamos la tecla C, presenta el número al cual va a comparar, si el número de conteo es igual a 24, activa un aviso auditivo (buzzer activo), este buzzer trabaja a 12 voltios, lo que le diferencia de los parlantes comunes es que no necesita ser activado con una frecuencia, si no basta con alimentarle con 12 voltios para que suene, también lo conocen con el nombre de chicharra a 12 voltios.

MATERIALES.

- 4 DISPLAYS ánodo común
- 4 transistores 2N3906
- 1 transistor 2N3904
- 7 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 4 resistencias de $4,7 K\Omega$ a $\frac{1}{2}$ vatio, amarillo-violeta-rojo
- 1 CI. 7447 decodificador BCD
- 2 pulsadores normalmente abiertos
- 1 chicharra de 12 V. como el de la figura 5.4.5.1.

El siguiente es el programa para controlar los 4 displays, contar, encerar, comparar y visualizar el número almacenado en la memoria.

```

unid  VAR BYTE           ;variable unidades
dece  VAR BYTE           ;variable decenas
cent  VAR BYTE           ;variable centenas
mile  VAR BYTE           ;variable miles

setunid  VAR BYTE       ;variable setunidades
setdece  VAR BYTE       ;variable setdecenas
setcent  VAR BYTE       ;variable setcentenas
setmile  VAR BYTE       ;variable setmiles

setunid=4           ;# que queremos que nos de aviso 0024 podemos cambiarlo
setdece=2           ;# que queremos que nos de aviso 0024 podemos cambiarlo
setcent=0           ;# que queremos que nos de aviso 0024 podemos cambiarlo
setmile=0           ;# que queremos que nos de aviso 0024 podemos cambiarlo

chicha VAR porta.0      ;pin de la chicharra
contar  VAR porta.3      ;pulsos para contar
encera  VAR porta.2      ;tecla encerar
visual  VAR porta.1      ;visualizar el valor a comparar
activar VAR BIT          ;bandera para la tecla contar

trish=0             ;todo el puerto b es de salida
cmcon=7             ;Todo el puerto A en modo digital

HIGH chicha         ;encendemos la chicharra para asegurarnos que el PIC está funcionando
PAUSE 200

```

continúa

```

encerar:
    unid=0                ;carga la variable unid con cero
    dece=0                ;carga la variable dece con cero
    cent=0                ;carga la variable cent con cero
    mile=0                ;carga la variable mile con cero
    LOW chicha            ;apagar la chicharra
display:
    portb= 224+unid       ;224 %111100000, activa las unidades
    PAUSE 5
    portb= 208+dece       ;208 %11010000, activa las decenas
    PAUSE 5
    portb= 176+cent       ;176 %10110000, activa las centenas
    PAUSE 5
    portb= 112+mile       ;112 %01110000, activa los miles
    PAUSE 5
    GOSUB teclas          ;revisar el estado de las teclas
    GOTO display
teclas:
    IF contar=0 THEN sumar ;si presionan tecla A ir a sumar
    IF encera=0 THEN encerar ;si presionan tecla B ir a encerar
    IF visual=0 THEN visualizar ;si presionan tecla C ir a visualizar
    activar=1
    RETURN                ;bandera de tecla A evita que cuente + de 1 vez
sumar:
    IF activar=0 THEN RETURN ; bandera de tecla A
    activar=0              ;bandera de tecla A cuando ya ha sido pulsada
    unid=unid+1            ;sumar 1 a las unidades
    IF unid<10 THEN comparar ;si unid es menor a 10 comparar
    unid=0                 ;hace cero a las unidades
    dece=dece+1            ;y incrementa las decenas
    IF dece<10 THEN comparar
    dece=0
    cent=cent+1
    IF cent<10 THEN comparar
    cent=0
    mile=mile+1
    IF mile<10 THEN comparar
    mile=0
    RETURN                ;retornar a gosub teclas
visualizar:
    portb= 224+setunid     ;224 %111100000,activa las unidades
    PAUSE 15
    portb= 208+setdece     ;208 %11010000,activa las decenas
    PAUSE 15
    portb= 176+setcent     ;176 %10110000,activa las centenas
    PAUSE 15
    portb= 112+setmile     ;112 %01110000,activa los miles
    PAUSE 15
    IF visual=0 THEN visualizar
    RETURN                ;retornar a gosub teclas

```

continúa

comparar:	
IF unid!=setunid THEN RETURN	;si unid no es igual a setunid
IF dece!=setdece THEN RETURN	;si dece no es igual a setdece
IF cent!=setcent THEN RETURN	
IF mile!=setmile THEN RETURN	
 HIGH chicha	;encender la chicharra
RETURN	;retornar a gosub teclas
END	

Figura 5.4.5.2. contador 1-9999.bas Programa para el contador decimal desde 1 a 9999.

A continuación otra manera de escribir el mismo programa aún más corto, esta vez utilizando una variable con capacidad de almacenamiento de 65535 (número VAR WORD), y para poder tomar cada dígito de esta variable y mostrar en cada uno de los displays, utilizamos el operador matemático DIG, que sirve par tomar cualquier dígito que necesitemos de una variable ejemplo:

Para tomar las decenas de la siguiente variable número:

número = 6789

cent= número DIG 1 , como el dígito 0 es el 9, el dígito 1 es el 8, el dígito 2 es 7, y así sucesivamente, en este caso el 8 se almacena en la variable cent.

numero VAR WORD	;creamos numero para almacenar el conteo
compara VAR WORD	;creamos compara para guardar el límite
unid VAR BYTE	;variable unidades
dece VAR BYTE	;variable decenas
cent VAR BYTE	;variable centenas
mile VAR BYTE	;variable miles
chicha VAR porta.0	;pin de la chicharra
contar VAR porta.3	;pulsos para contar
encera VAR porta.2	;tecla encerrar
visual VAR porta.1	;visualizar el valor a comparar
activar VAR BIT	;bandera para la tecla contar
 trish=0	;todo el puerto b es de salida
cmcon=7	;Todo el puerto A en modo digital
HIGH chicha	;encendemos la chicharra para asegurarnos que
PAUSE 200	;el PIC está funcionando
compara=12	;el número al cual va a comparar es el 12
 encerrar:	
numero=0	;carga la variable número con cero
LOW chicha	;apagar la chicharra
display:	
unid=numero DIG 0	;toma el dígito 0 (unidades) y guarda en unid
dece=numero DIG 1	;toma las decenas y lo guarda en dece
cent=numero DIG 2	;toma el dígito 2 (centenas) y lo guarda en cent
mile=numero DIG 3	;toma el dígito 3 (miles) y lo guarda en mile

continúa


```

portb= 224+unid                ;224 %11100000,activa las unidades
PAUSE 5
portb= 208+dece                ;208 %11010000,activa las decenas
PAUSE 5
portb= 176+cent                ;176 %10110000,activa las centenas
PAUSE 5
portb= 112+mile                ;112 %01110000,activa los miles
PAUSE 5
GOSUB teclas                    ;revisar el estado de las teclas
GOTO display

teclas:
IF contar=0 THEN sumar         ;si presionan tecla A ir a sumar
IF encera=0 THEN encerar      ;si presionan tecla B ir a encerar
IF visual=0 THEN visualizar    ;si presionan tecla C ir a visualizar
activar=1                      ;bandera de tecla A evita que cuente + de 1 vez
RETURN

sumar:
IF activar=0 THEN RETURN       ;bandera de tecla A
activar=0                      ;bandera de tecla A cuando ya ha sido pulsada
numero=numero+1               ;sumar 1
IF numero=compara THEN HIGH chicha ;si número es = compara
IF numero>9999 THEN encerar    ;si el número es >9999 ir a encerar
RETURN                         ;retornar a gosub teclas

visualizar:
unid=compara DIG 0             ;toma el dígito 0 (unidades) y guarda en unid
dece=compara DIG 1             ;toma el dígito 1 (decenas) y lo guarda en dece
cent=compara DIG 2             ;toma las centenas y lo guarda en cent
mile=compara DIG 3             ;toma el dígito 3 (miles) y guarda en mile
portb= 224+unid                ;224 %11100000,activa las unidades
PAUSE 15
portb= 208+dece                ;208 %11010000,activa las decenas
PAUSE 15
portb= 176+cent                ;176 %10110000,activa las centenas
PAUSE 15
portb= 112+mile                ;112 %01110000,activa los miles
PAUSE 15
IF visual=0 THEN visualizar
RETURN                         ;retornar a gosub teclas

END

```

Figura 5.4.5.3. contador2 1-9999.bas Programa para el contador decimal desde 1 a 9999 utilizando el operador matemático DIG.

5.4.6. MANEJO DE 4 DISPLAYS DE 7 SEGMENTOS SIN EL CI. 7447 (ROTULACIÓN).

Es posible hacer un contador decimal igual que el del proyecto anterior y sin ayuda del CI. 7447,

pero en esta ocasión haremos algo más que eso, como vieron la ventaja de conectar el display directamente al PIC es la de poder sacar casi la mayoría de las letras del alfabeto, pues bien este proyecto consiste en sacar la palabra HOLA a través de los 4 displays.

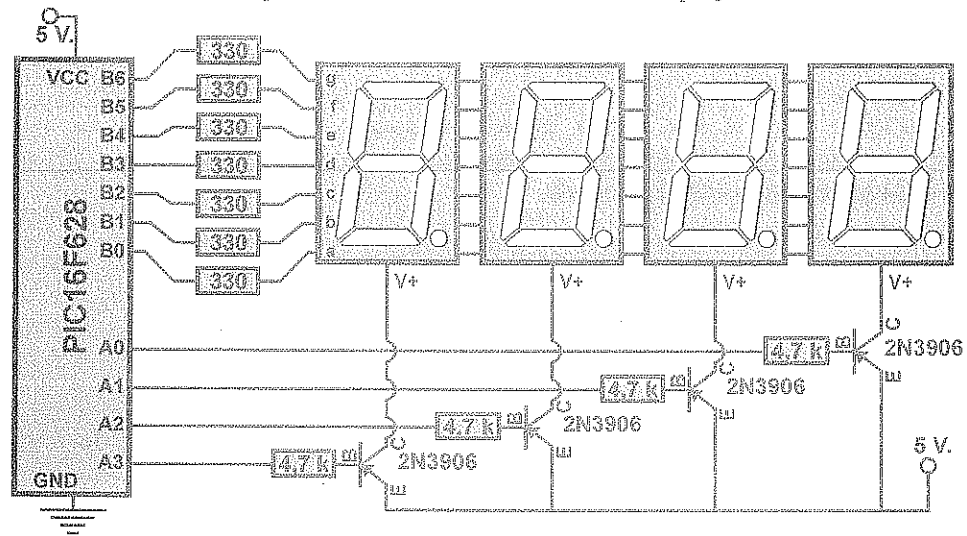


Figura 5.4.6.1. Esquema de conexionado, para manejar 4 displays directamente desde el PIC y multiplexando por el puerto A.

MATERIALES.

- 4 DISPLAYS ánodo común
- 4 transistores 2N3906
- 7 resistencias de 330Ω a ½ vatio, naranja-naranja-café
- 4 resistencias de 4,7 KΩ a ½ vatio, amarillo-violeta-rojo

```

cmcon=7           ;convierte todo el puerto A en digital
trish=0           ;convierte todos los pines del puerto B en salidas
trisa=0           ;convierte todos los pines del puerto A en salidas
texto:
    porta=14       ;%1110 activa el display de la derecha
    portb=8        ;%0001000 forma la letra A
    PAUSE 5
    porta=13       ;%1101 activa el siguiente display
    portb=71       ;%1000111 forma la letra L
    PAUSE 5
    porta=11       ;%1011 activa el siguiente display
    portb=64       ;%1000000 forma la letra O
    PAUSE 5
    porta=7        ;%0111 activa el último display
    portb=9        ;%0001001 forma la letra H
    PAUSE 5
    GOTO texto
    END
    
```

Figura 5.4.6.2. palabra HOLA 4D.bas Programa para presentar la palabra HOLA en los 4 displays.

5.4.7. MANEJO DE 4 DISPLAYS COMO RÓTULO EN MOVIMIENTO.

Para hacerlo más interesante el proyecto anterior le añadiremos movimiento, con esto podemos ingresar frases completas como "HOLA LUIS", pero para no alargar mucho el programa solo utilizaremos la palabra HOLA, moviéndose continuamente de derecha a izquierda y con un espacio por palabra.

La variable x es la que regula la velocidad con que se desplazan las letras, para comprobarlo modifique el valor de todas las repeticiones de 1 TO 20 al doble 1 TO 40, y verá como se desplazan las letras más lentamente.

```

emcon=7           ;convierte todo el puerto A en digital
trisb=0           ;convierte todos los pines del puerto B en salidas
trisa=0           ;convierte todos los pines del puerto A en salidas
x VAR BYTE        ;crea la variable x con un tamaño de 255

texto:
FOR x=1 TO 20      ;repeticiones de este segmento
  porta=14 :portb=8 ;pA%1110 y pB%0001000 forma la letra A
  PAUSE 5
  porta=13 :portb=71 ;pA%1101 y pB%1000111 forma la letra L
  PAUSE 5
  porta=11 :portb=64 ;pA%1011 y pB%1000000 forma la letra O
  PAUSE 5
  porta=7 :portb=9 ;pA%0111 y pB%0001001 forma la letra H
  PAUSE 5
NEXT
FOR x=1 TO 20      ;repeticiones de este segmento
  porta=14 :portb=127 ;pA%1110 y pB%1111111 apaga el display
  PAUSE 5
  porta=13 :portb=8 ;pA%1101 y pB%0001000 forma la letra A
  PAUSE 5
  porta=11 :portb=71 ;pA%1011 y pB%1000111 forma la letra L
  PAUSE 5
  porta=7 :portb=64 ;pA%0111 y pB%1000000 forma la letra O
  PAUSE 5
NEXT
FOR x=1 TO 20      ;repeticiones de este segmento
  porta=14 :portb=9 ;pA%1110 y pB%0001001 forma la letra H
  PAUSE 5
  porta=13 :portb=127 ;pA%1101 y pB%1111111 apaga el display
  PAUSE 5
  porta=11 :portb=8 ;pA%1011 y pB%0001000 forma la letra A
  PAUSE 5
  porta=7 :portb=71 ;pA%0111 y pB%1000111 forma la letra L
  PAUSE 5
NEXT
FOR x=1 TO 20      ;repeticiones de este segmento
  porta=14 :portb=64 ;pA%1110 y pB%1000000 forma la letra O
  PAUSE 5
  porta=13 :portb=9 ;pA%1101 y pB%0001001 forma la letra H

```

continúa


```

PAUSE 5
porta=11 :portb=127      ;pA%1011 y pB%11111111 apaga el display
PAUSE 5
porta=7 :portb=8         ;pA%0111 y pB%0001000 forma la letra A
PAUSE 5
NEXT
FOR x=1 TO 20            ; repeticiones de este segmento
porta=14 :portb=71      ;pA%1110 y pB%1000111 forma la letra L
PAUSE 5
porta=13 :portb=64      ;pA%1101 y pB%1000000 forma la letra O
PAUSE 5
porta=11 :portb=9       ;pA%1011 y pB%0001001 forma la letra H
PAUSE 5
porta=7 :portb=127      ;pA%0111 y pB%11111111 apaga el display
PAUSE 5
NEXT
GOTO texto
END

```

Figura 5.4.7.1. p HOLA moviendo.bas Programa para desplazar la palabra HOLA.

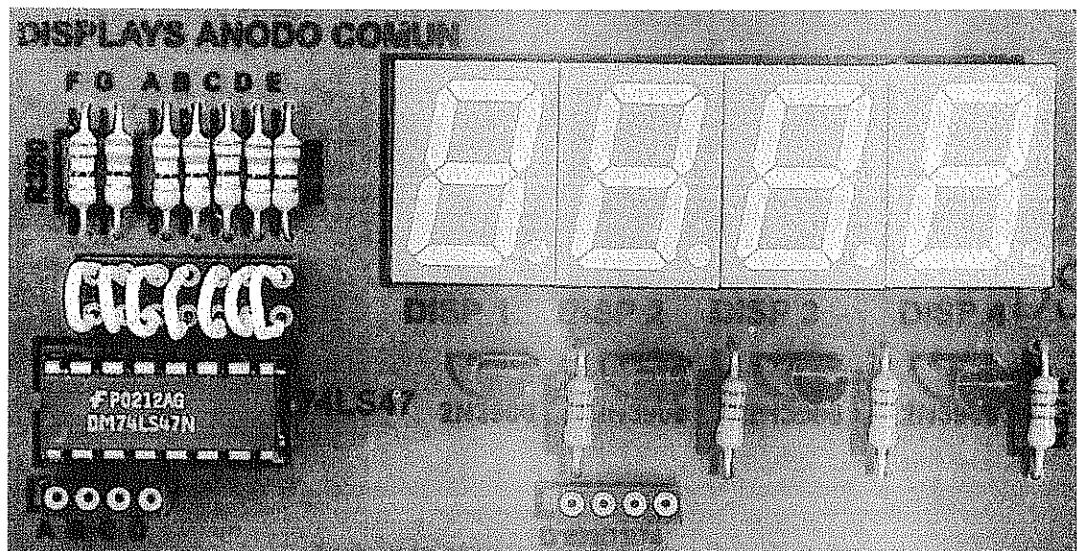


Figura 5.4.7.2. Fotografía del bloque de 4 displays a. c. del entrenador experto EE 01.

5.4.8. MANEJO DE UN DISPLAY DE 35 SEGMENTOS.

Estos displays son muy utilizados para transmitir mensajes en bancos y instituciones de atención al público, la ventaja de estos son la gran cantidad de caracteres que se pueden formar y el gran tamaño de los displays (5,5 x 4 cm), en esta práctica aprenderemos a manejar un display de 35 segmentos de 28 pines el cual es bicolor (rojo y verde), pero también existen displays de un solo

color y tienen solo 14 pines. Esta práctica consiste en formar un hombrecito saludando, una vez familiarizado, será muy sencillo ir implementando más displays del mismo tipo.

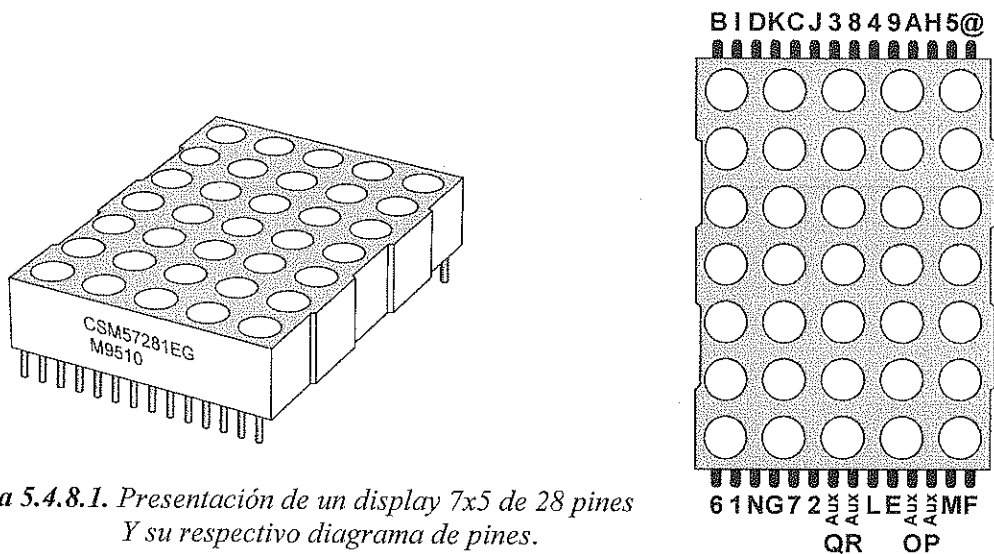


Figura 5.4.8.1. Presentación de un display 7x5 de 28 pines y su respectivo diagrama de pines.

Ejemplo: si queremos encender el 1er led en color rojo, debemos conectar A a 5 V. y 1 a Gnd, y si queremos en color verde H y 6, también podemos unir los pines 1 y 6 ya que los dos son tierra y trabajar solo uniendo H y A a positivo, esto nos simplificará el circuito.

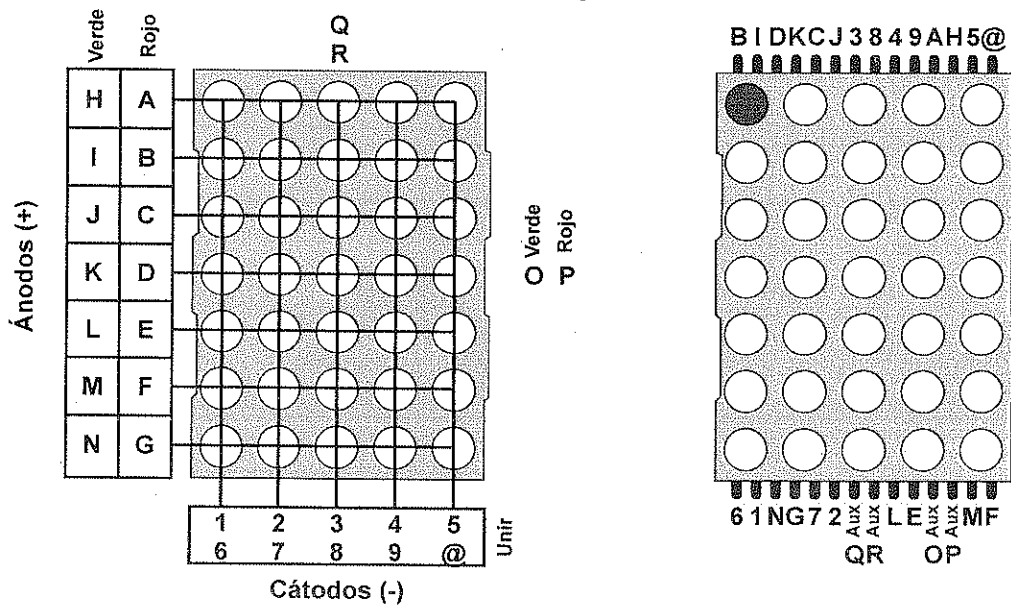


Figura 5.4.8.2. Esquema de matrices de un display de 35 segmentos, y encendido del 1er LED en color rojo, conectando el pin A a positivo y 1 a tierra.

MATERIALES.

- 1 DISPLAY 7 x 5
- 5 transistores 2N3904
- 7 resistencias de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 5 resistencias de $4,7 K\Omega$ a $\frac{1}{2}$ vatio, amarillo-violeta-rojo.

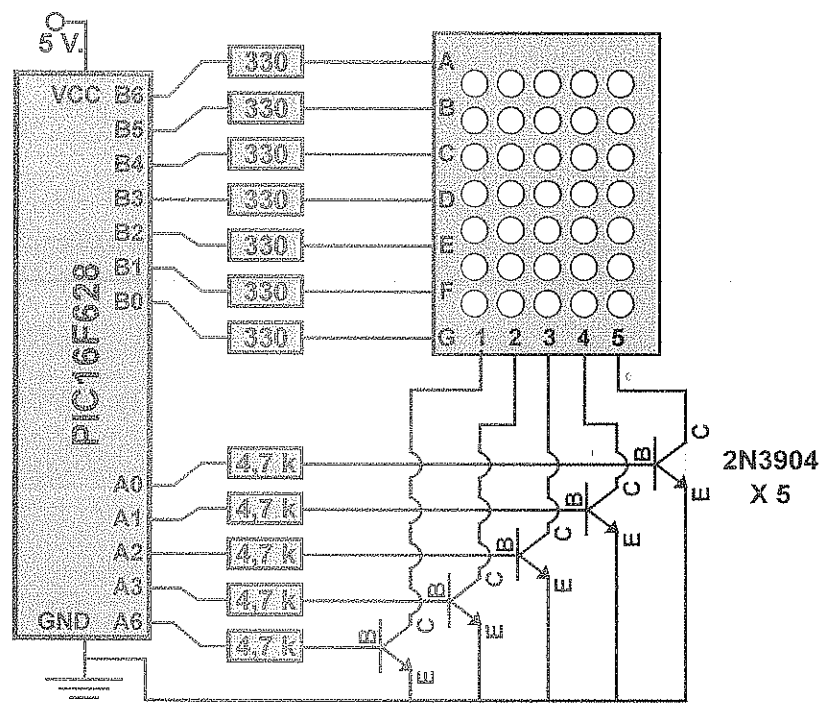


Figura 5.4.8.3. Esquema de conexión de un display 7 x 5 a un PIC.

NOTA: En esta práctica se utilizó transistores 2N3904, pero si vamos a utilizar más displays 7x5 y queremos mayor iluminación, se recomienda reemplazarlos por transistores TIP110 y adicionalmente colocar en la salida del puerto B transistores 2N3906, con una resistencia limitadora de 22 Ω a voltaje positivo. (ver figura 5.4.8.6.)

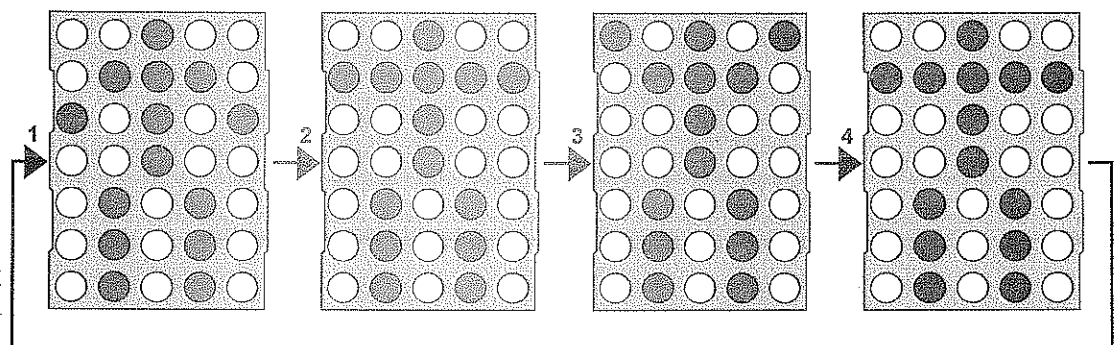


Figura 5.4.8.4. Secuencia de la animación del hombrecito saludando.

El programa a realizar, debe multiplexar los leds de forma que se encienda como la secuencia 1, permanece 100 milisegundos y cambia a la secuencia 2, luego a la secuencia 3, y finalmente a la secuencia 4, para luego volver a repetir toda la secuencia desde el principio. Como resultado observaremos un hombrecito que agita los brazos, pruebe con otras figuras y letras.

cmcon=7	;convierte el puerto A en digital
trisa=0	;hace salida todo el puerto A
trisb=0	;convierte en salidas el puerto B
x VAR BYTE	;crea variable x de 255
animacion:	
FOR x = 1 TO 10	;repetir esta secuencia 10 veces
porta=%0000001 :portb=%0010000 :PAUSE 4	;esperar 4 mls, total 20 mls cada
porta=%0000010 :portb=%0100111 :PAUSE 4	;escena
porta=%0000100 :portb=%1111000 :PAUSE 4	
porta=%0001000 :portb=%0100111 :PAUSE 4	
porta=%1000000 :portb=%0010000 :PAUSE 4	
NEXT	
FOR x = 1 TO 10	;repetir esta secuencia 10 veces
porta=%0000001 :portb=%0100000 :PAUSE 4	
porta=%0000010 :portb=%0100111 :PAUSE 4	
porta=%0000100 :portb=%1111000 :PAUSE 4	
porta=%0001000 :portb=%0100111 :PAUSE 4	
porta=%1000000 :portb=%0100000 :PAUSE 4	
NEXT	
FOR x = 1 TO 10	;repetir esta secuencia 10 veces
porta=%0000001 :portb=%1000000 :PAUSE 4	
porta=%0000010 :portb=%0100111 :PAUSE 4	
porta=%0000100 :portb=%1111000 :PAUSE 4	
porta=%0001000 :portb=%0100111 :PAUSE 4	
porta=%1000000 :portb=%1000000 :PAUSE 4	
NEXT	
FOR x = 1 TO 10	;repetir esta secuencia 10 veces
porta=%0000001 :portb=%0100000 :PAUSE 4	
porta=%0000010 :portb=%0100111 :PAUSE 4	
porta=%0000100 :portb=%1111000 :PAUSE 4	
porta=%0001000 :portb=%0100111 :PAUSE 4	
porta=%1000000 :portb=%0100000 :PAUSE 4	
NEXT	
GOTO animacion	;ir a animación
END	

Figura 5.4.8.5. hombrecito.bas Programa para animación.

Como podemos observar en el programa anterior, tratamos de no utilizar el puerto A.4 y A.5, porque el primero es de colector abierto, y el segundo solo puede trabajar como entrada, además este programa se puede reducir por lo menos a la mitad si utilizamos **GOSUB** en la secuencia del hombre con los brazos en la mitad, ya que este se repite 2 veces, también en los **PAUSE 4** y algunas partes de secuencias como las piernas y la cabeza que no se mueven, estos podemos agrupar también en una subrutina de **GOSUB**, pero para poder entender bien el funcionamiento se prefirió no reducir el programa.

Es muy importante considerar el tiempo de multiplexaje, ya que no se debe sobrepasar un total de 20 mls, por ejemplo si utilizamos 2 displays 7x5, debemos bajar el tiempo de pausas a 2 milisegundos para también tener un total de 20 mls.

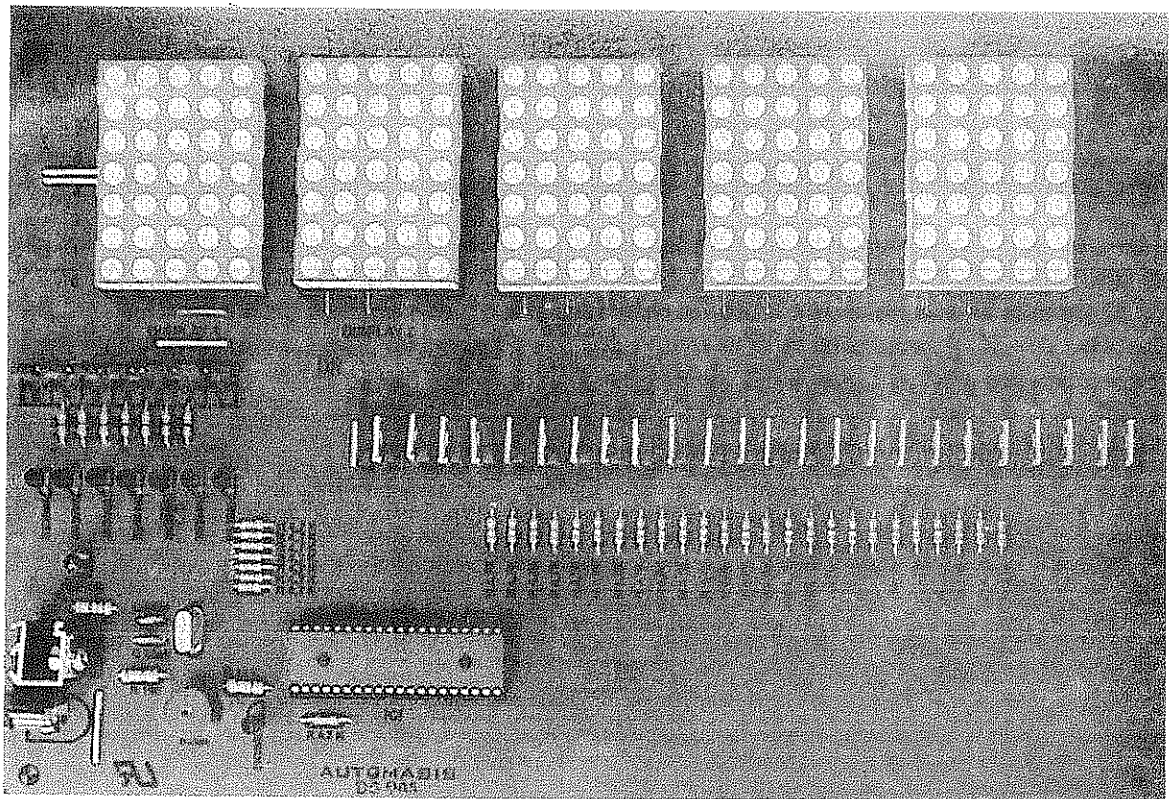


Figura 5.4.8.6. Fotografía de una placa con 5 displays 7x5, comandado por un PIC16F877, noten que los transistores 2N3904 fueron reemplazados por transistores TIP110, con la finalidad de conseguir mayor iluminación en los leds.

5.5 MÓDULOS LCD

5.5.1. MANEJO DE UN MÓDULO LCD.

Los módulos LCD (Display de Cristal Líquido), son utilizados para mostrar mensajes que indican al operario el estado de la maquina, o para dar instrucciones de manejo, mostrar valores, etc. El LCD permite la comunicación entre las máquinas y los humanos, este puede mostrar cualquier caracter ASCII, y consumen mucho menos que los displays de 7 segmentos, existen de varias presentaciones por ejemplo de 2 líneas por 8 caracteres, 2x16, 2x20, 4x20, 4x40, etc. sin backlight (14 pines) o con backlight (16 pines, iluminado de pantalla), el LCD más popular es el 2x16, 2 líneas de 16 caracteres cada una.

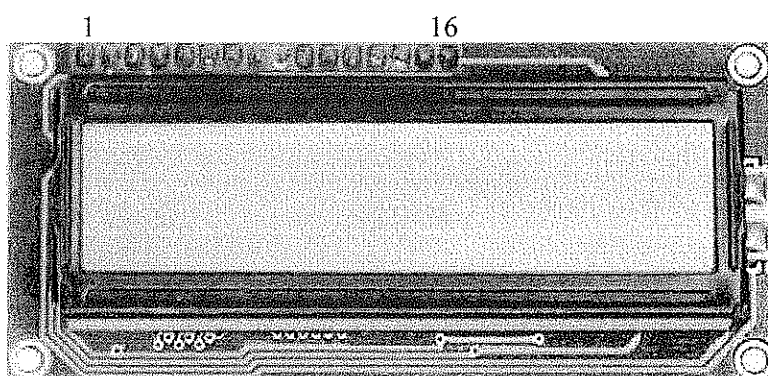


Figura 5.5.1.1. Fotografía de un LCD 2x16 con controlador Hitachi 44780 y Backlight en color amarillo

Pin	Simb.	Descripción
1	Vss	Tierra de alimentación GND
2	Vdd	Alimentación de +5V CC
3	Vo	Ajuste del contraste del cristal líquido (0 a +5V)
4	RS	Selección del registro control/datos RS=0 reg. control RS=1 reg. datos
5	R/W	Lectura/escritura en LCD R/W=0 escritura (Write) R/W=1 lectura (Read)
6	E	Habilitación E=0 módulo desconectado E=1 módulo conectado
7	D0	Bit menos significativo (bus de datos bidireccional)
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	Bit más significativo (bus de datos bidireccional)
15	A	Alimentación del backlight +3,5 V o +5V CC (según especificación técnica)
16	K	Tierra GND del backlight

Figura 5.5.1.2. Función de cada pin del LCD.

LA DECLARACIÓN LCDOUT. Sirve para mostrar items en una pantalla de cristal líquido, se utiliza escribiendo: LCDOUT, luego escribiendo \$FE, y seguido por el comando a utilizar, el siguiente cuadro muestra los comandos más utilizados:

Comando	Operación
\$FE, 1	Limpia el visor del LCD
\$FE, 2	Vuelve al inicio (comienzo de la primera línea)
\$FE, \$0C	Apagar el cursor
\$FE, \$0E	Subrayado del cursor activo (—)
\$FE, \$0F	Parpadeo del cursor activo (■)
\$FE, \$10	Mover el cursor una posición a la izquierda
\$FE, \$14	Mover el cursor una posición a la derecha
\$FE, \$80	Mueve el cursor al comienzo de la primera línea
\$FE, \$C0	Mueve el cursor al comienzo de la segunda línea
\$FE, \$94	Mueve el cursor al comienzo de la tercera línea
\$FE, \$D4	Mueve el cursor al comienzo de la cuarta línea

Figura 5.5.1.3. Tabla de los comandos más utilizados para manejar un LCD.

Los LCD se puede conectar con el PIC con un bus de 4 u 8 bits, la diferencia está en el tiempo que se demora, pues la comunicación a 4 bits, primero envía los 4 bits más altos y luego los 4 bits más bajos, mientras que la de 8 bits envía todo al mismo tiempo, esto no es un inconveniente si consideramos que el LCD trabaja en microsegundos. Pero la gran ventaja de hacer conexión a 4 bits, son los pocos cables que se deben conectar, como podemos ver en la figura 5.5.1.4. solo debemos conectar el bit de Registro, el Enable y los 4 bits más altos del LCD, con esto es suficiente para enviar los mensajes.

El compilador PBP soporta módulos LCD'S con controlador Hitachi 44780 o equivalentes y por defecto, asume que se conectó en el pin A4 el bit de Registro, en el pin B3 el bit Enable y en el puerto A empezando desde A0 hasta A3, los bits más altos del LCD. Esta configuración predefinida, se lo puede cambiar de acuerdo a la necesidad, como lo veremos más adelante.

MATERIALES.

- 1 DISPLAY LCD 2x16 (preguntar si es compatible con los PIC'S)
- 1 resistencia de 10Ω a $\frac{1}{2}$ watio, café-negro-negro
- 1 potenciómetro de 10 K Ω .

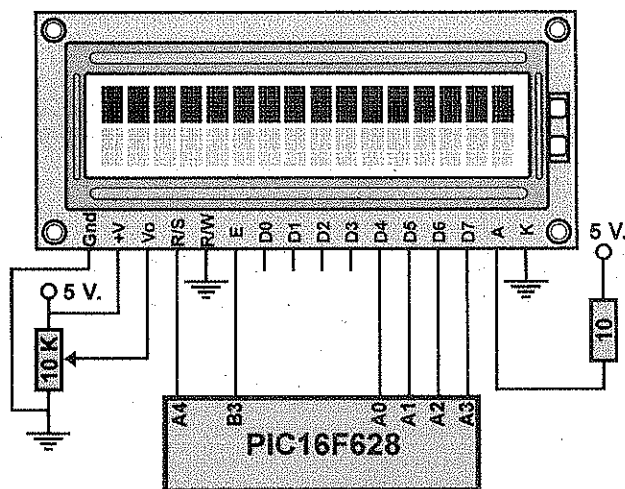


Figura 5.5.1.4. Conexión de un LCD, a 4 bits predefinido por el compilador PBP, la resistencia de 10Ω conectado a la alimentación del backlight, sirve para evitar altas temperaturas, noten además que el bit R/W se encuentra conectado a tierra, esto es porque la declaración LCDOUT, es de escritura únicamente.

PAUSE 200	;retardo para esperar que funcione el LCD
LCDOUT \$FE, 1,"Hola"	;limpiar pantalla y sacar el texto Hola
LCDOUT \$FE, \$C0,"microPIC"	;pasar al comienzo de la segunda línea y escribir microPIC
END	;fin de instrucciones

Figura 5.5.1.5. *LCD Hola.bas Programa para presentar la palabra Hola microPIC.*

NOTA: algunos LCD'S no requieren de ningún PAUSE al inicio, pero existen otros modelos que necesitan unos pocos milisegundos para estar listos, por eso colocamos un **PAUSE 200** al comienzo del programa.

Bien una vez visto el texto notaremos que las dos palabras están al lado izquierdo, si queremos que salgan centradas en nuestro LCD, tenemos 2 maneras de hacerlo, la primera es dando espacios antes de cada palabra ejemplo:

LCDOUT \$FE, 1, " Hola" y LCDOUT \$FE, \$C0, " microPIC"

Lo cual es sencillo pero no es muy recomendable porque ocupa más espacio en el PIC, la segunda manera es asignando el lugar donde se quiere que aparezca cada palabra ejemplo:

LCDOUT \$FE, 1	;limpia la pantalla y coloca el cursor al comienzo
LCDOUT \$FE, \$86, "Hola"	;pasa el cursor al 7ma casilla de la 1era Línea y escribe
LCDOUT \$FE, \$C4, "microPIC"	;pasa a la casilla 5 de la 2da línea y escribe microPIC

Se debe entender que existe un cursor que aunque no lo vemos, pues este es el que indica donde aparecerá la siguiente letra, para poder entender haremos un ejercicio completo, así podrán aprender más del LCD y las funciones de cada uno de los comandos. Primero que nada haremos visible el cursor y luego pondremos PAUSES para poder ver el funcionamiento.

x VAR BYTE	;crea la variable x de un tamaño de 255
pepe CON \$FE	;asigna el nombre de pepe a la constante \$FE
 PAUSE 2000	
LCDOUT pepe, 1	;limpia el visor del LCD
PAUSE 2000	
LCDOUT pepe,\$0F	;muestra el cursor en casilla negra
PAUSE 2000	
LCDOUT pepe,\$0E	;subraya el cursor
PAUSE 2000	
LCDOUT pepe,\$14	;desplaza el cursor una casilla a la derecha
PAUSE 2000	
LCDOUT , "MIKRO"	;escribe mikro, desde donde se encuentre el cursor
PAUSE 2000	
FOR x = 1 TO 3	; repite 3 veces las siguientes instrucciones
LCDOUT pepe,\$10	; desplaza el cursor una casilla a la izquierda
PAUSE 1000	
NEXT	
LCDOUT , 67	; envía el caracter ASCII "C" para corregir MICRO
PAUSE 2000	

continúa

LCDOUT pepe,\$C0+12,"PIC"	;escribe en la segunda línea casillero 13
	;esto equivale a \$CC o 204
PAUSE 2000	
LCDOUT pepe,2,"1"	;vuelve al inicio de la 1era fila y escribe 1
END	

Figura 5.5.1.6. *LCD Hola2.bas Programa utilizando la mayoría de los comandos del LCD.*

Observen que la constante \$FE se le cambió por pepe, así mismo si se les dificulta memorizar como pasar a la segunda línea, puede definir la constante: **lin2 CON \$C0**, y cuando deseen escribir en la segunda línea pondrían: **LCDOUT** pepe, lin2, "hola", o lo que es lo mismo utilizando números decimales: **LCDOUT** 254, 192, "hola". También cabe recalcar que el LCD tiene una memoria RAM (Random Access Memory) que lo explicaremos más adelante, por lo que una vez que se le envía el texto, este permanece ahí y el PIC se lo puede utilizar para otras tareas o podemos desconectarlo si lo deseamos.

En ocasiones especiales se debe cambiar la configuración de los pines del PIC hacia el LCD, por ejemplo para utilizar los comparadores de voltaje que se encuentran en el puerto A, necesitamos dejar disponibles estos pines, esto se logra adicionando al principio lo siguiente:

DEFINE LCD_DREG	PORTB	; define pines del LCD B4 a B7
DEFINE LCD_DBIT	4	; empezando desde el Puerto B4 hasta el B7
DEFINE LCD_RSREG	PORTB	;define pin para conectar el bit RS
DEFINE LCD_RSBIT	3	;en el puerto B3
DEFINE LCD_EREG	PORTB	;define pin para conectar el bit Enable
DEFINE LCD_EBIT	2	;en el puerto B2

Una vez que se define la nueva configuración de pines para el LCD, programamos de la misma forma que las ocasiones anteriores, es importante además saber que los 4 bits de datos solo se pueden configurar en los 4 bits más bajos (B.0 al B.3) o los 4 bits más altos (B.4 al B.7) de un puerto del PIC, y si deseamos hacer una comunicación a 8 bits con el LCD, estos deben estar en un solo puerto, además debemos definir en el PBP que vamos a utilizar un bus de 8 bits, esto es de la siguiente manera:

DEFINE LCD_BITS 8 ; define comunicación a 8 bits con el LCD

Y si nuestro LCD posee 4 líneas, también debemos definirlo de la siguiente forma:

DEFINE LCD_LINES 4 ; define un LCD de 4 líneas

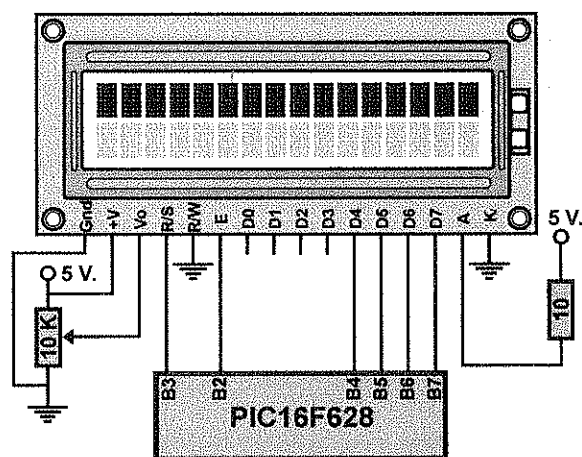
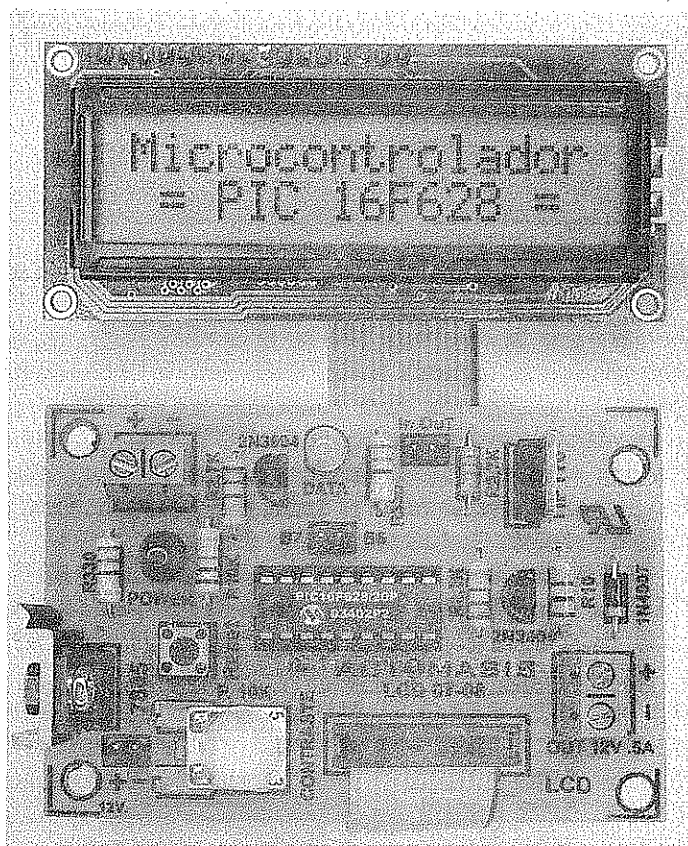


Figura 5.5.1.7. *Conexión de un LCD a 4 bits con una configuración diferente a la predefinida por el PBP, esta vez utilizando el puerto B3 para el bit R/S el B2 para el Enable y desde el B4 al B7 para los bits de comunicación.*

Figura 5.5.1.8. Fotografía de una tarjeta que controla un LCD 2x16 el cual posee un PIC16F628 que se encarga de enviar los datos y además enciende el backlight a través de un transistor 2N3904 dispone además:

*Un led que indica envío de datos
Una salida de transistor TIP110
Una salida de transistor 2N3904
Un pulsador de reset al MCLR
Un potenciómetro para el ajuste del contraste.*



5.5.2. PRESENTACIÓN DE CARACTER POR CARACTER EN LCD.

En las prácticas anteriores se presentaron mensajes completos en un instante, en esta nueva práctica incluimos la declaración **LOOKUP**, que nos servirá para enviar caracter por caracter con un intervalo de 400ms, dando como resultado un efecto especial en la visualización. Como conexión para esta práctica utilizaremos la misma de la figura 5.5.1.4.

PAUSE 200	;retardo para esperar que funcione el LCD
x VAR BYTE	;crear la variable x de 255
abc VAR BYTE	;crear la variable abc de 255
ini:	
LCDOUT \$FE,1	;limpiar pantalla
FOR x = 0 TO 15	;repetir 16 veces
LOOKUP x,["Microcontrolador"],abc	;tomar caracter por caracter y guardar en abc
LCDOUT , abc	;sacar en LCD el contenido de abc
PAUSE 400	;esperar 400 mls
NEXT	;siguiente repetición
PAUSE 2000	
GOTO ini	
END	

Figura 5.5.2.1. LCD especial.bas Programa para mostrar uno por uno cada caracter.

5.5.3. DESPLAZAMIENTO DE UN TEXTO EN LCD.

El LCD dispone en cada una de las líneas 40 posiciones de memoria, de los cuales únicamente 16 son visibles, en el siguiente ejercicio escribiremos un mensaje desde el casillero 17 (\$90), el cual no es visible y luego iremos desplazando a la izquierda, como resultado tendremos un texto que se mantiene en movimiento, una vez que este termina recorrerá 16 posiciones en blanco y luego volverá a aparecer los 24 caracteres del texto.

PAUSE 200	;retardo para esperar que funcione el LCD
x VAR BYTE	;crear la variable x de 255
abc VAR BYTE	;crear la variable abc de 255
LCDOUT \$FE,\$7	;configura para desplazamiento izquierdo
LCDOUT \$FE,1	;limpiar pantalla
ini:	
LCDOUT \$FE,\$90	;ubica el cursor en la casilla 17
FOR x = 0 TO 23	;repetir 24 veces
LOOKUP x,["Microcontroladores -PIC-"],abc	;tomar caracter por caracter y guardar en abc
LCDOUT , abc	;sacar en LCD el contenido de abc
PAUSE 400	;esperar 400 mls
NEXT	;siguiente repetición
GOTO ini	
END	

Figura 5.5.3.1. LCD especial2.bas Programa para mostrar un texto en movimiento.

5.5.4. CONTADOR DE PULSOS CON LCD.

Este proyecto, consiste en contar el número de pulsos que ingresan por un pin en un determinado período, este a su vez se visualiza en un LCD, si la cantidad de este supera a los 120 pulsos por segundo es decir 120 HZ, se encenderá una alarma que en este caso será un led rojo, y si la cantidad de pulsos baja a menos de 100 HZ, este encenderá un led verde, si la frecuencia se mantiene entre estos 2 rangos, no se encenderá ningún led.

Este proyecto tiene muchas aplicaciones como por ejemplo para un regulador de voltaje en el que a más de indicarnos el voltaje de salida podría además indicarnos la frecuencia.

Para esta práctica utilizaremos un CI. 555 que nos ayuda a generar un tren de pulsos variable, el cual lo conectamos al PIC para su posterior conteo.

LA DECLARACIÓN COUNT. Sirve para contar el número de pulsos que ingresan por un pin en un determinado tiempo, este a su vez lo guarda en una variable para su posterior procesamiento, la manera de utilizarlo es la siguiente:

```
COUNT portb.0, 1000, abc
```

El cual se interpreta así: cuenta pulsos a través del puerto B0 en un período de 1000 milisegundos y lo guarda en la variable previamente creada llamada abc, el período podemos variarlo de 1 a 65535.

MATERIALES.

- 1 DISPLAY LCD 2x16
- 1 resistencia de 10Ω , y 1 resistencia de $1K\Omega$
- 3 leds de 5mm, 1 verde y 2 rojos
- 3 resistencia de 330Ω a $\frac{1}{2}$ vatio, naranja-naranja-café
- 2 potenciómetros de $10K\Omega$
- 1 CI. 555
- 1 capacitor de $10\mu F/25V$.

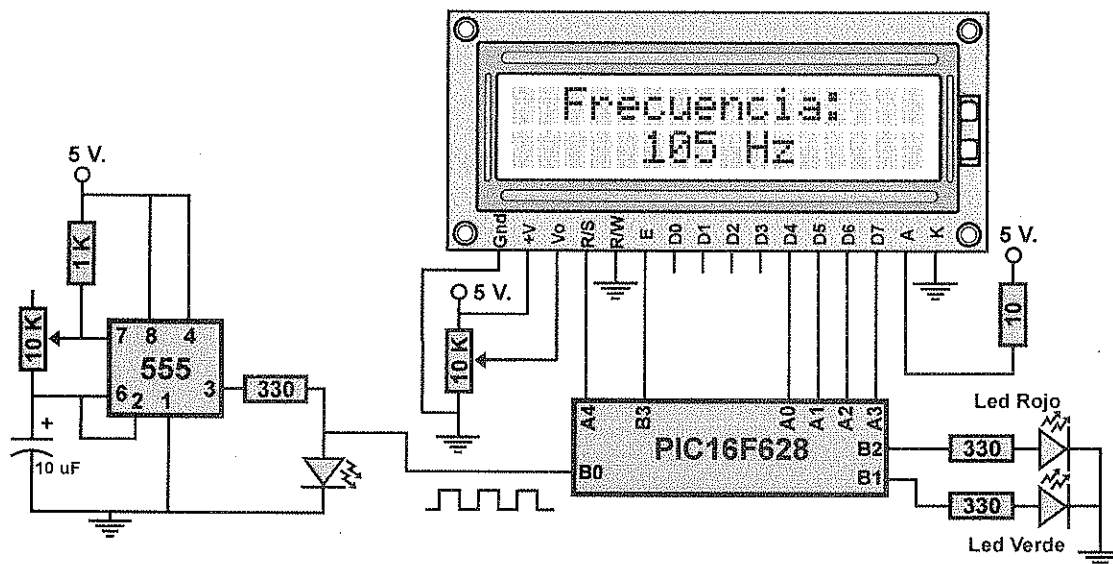


Figura 5.5.4.1. Esquema de conexión para el frecuencímetro con LCD.

```
CMCON = 7
LEDRED VAR PORTB.2
LEDVER VAR PORTB.1
puls VAR WORD
prog:
    COUNT portb.0,1000,puls
    LCDOUT SFE, 1
    LCDOUT SFE,$82,"Frecuencia:"
    LCDOUT SFE,$C5 ,DEC puls," Hz"

    IF puls>120 THEN alarmah
    IF puls<100 THEN alarmal
    LOW LEDRED : LOW LEDVER
    GOTO prog
alarmah: HIGH LEDRED
        LOW LEDVER
    GOTO prog
alarmal: HIGH LEDVER
        LOW LEDRED
    GOTO prog

;convierte en digitales el puerto A
;el puerto B.2 se llamará LEDRED
;el puerto B.2 se llamará LEDVER
;variable puls con capacidad de 0 a 65535

;contar pulsos en el puerto B.0
;limpiar pantalla LCD
;cursor en el casillero 3 de la 1era línea
;en el comienzo 2da línea casilla 6 sacar
;el valor decimal de la variable puls
;si la variable excede de 120
;si la variable baja de 100
;apagar los 2 leds

;encender led rojo
;apagar led verde

;encender led verde
;apagar led rojo
```

Figura 5.5.4.2. frecuenciaLCD.bas Programa para un contador de pulsos.

LA PALABRA DEC. Sirve para mostrar el número de la variable en decimal, también se lo puede representar por el signo (#), además existe las palabras **BIN** y **HEX**, el siguiente es un ejemplo de cómo mostraría el LCD si puls = 105:

LCDOUT \$FE, \$C5, DEC puls, " Hz" ; muestra en el LCD así : 105 Hz

También se lo puede utilizar el signo #, que equivale a **DEC**.

LCDOUT \$FE, \$C5, # puls, " Hz" ; muestra en el LCD así : 105 Hz

Si deseamos ver la variable en hexadecimal pondríamos así:

LCDOUT \$FE, \$C5, HEX puls, " Hz" ; muestra en el LCD así : 69 Hz

Y si queremos verlo en binario:

LCDOUT \$FE, \$C5, BIN puls, " Hz" ; muestra en el LCD así : 1101001 Hz

Si no colocamos ninguna instrucción nos mostraría el ASCII que representa el número 105, es decir la letra i.

LCDOUT \$FE, \$C5, puls, " Hz" ; muestra en el LCD así : i Hz

5.5.5. TACÓMETRO DIGITAL.

Este proyecto es muy similar al anterior, con la diferencia que el tren de pulsos ya no es generado por un C.I. 555, sino más bien por el giro de un motor que se une a un cono de caucho, el cual transmite movimiento a un disco de plástico negro, en el que posee una ranura de 1 a 2 mm, que al pasar por el medio del optoacoplador, polariza 2 transistores y este hace cambiar el estado de 0 a 1, esta señal podemos conectarlo al PIC y visualizarlo en un LCD, el mismo principio utilizan los marcadores de kilometraje de los autos y los tacómetros de los mismos.

MATERIALES.

- 1 DISPLAY LCD 2x16
- 1 resistencia de 10Ω
- 1 resistencia de 330Ω
- 1 resistencia de $1K\Omega$
- 1 resistencia de $4,7K\Omega$
- 1 potenciómetros de $10K\Omega$
- 1 eje de giro independiente con un disco de plástico negro
- 1 optoacoplador ECG3100 con salida de transistor NPN como el de la figura 5.5.5.2

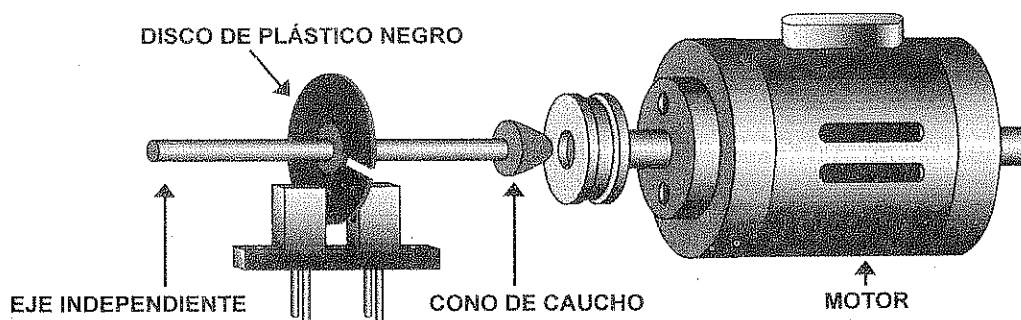


Figura 5.5.5.1. Esquema para unir el motor con el eje independiente de un tacómetro portátil.

En este caso no debemos multiplicar con ningún valor la variable revo, el inconveniente sería que deberíamos permanecer conectados al motor por un minuto, hasta que la variable **COUNT** termine de contar los pulsos, por esto lo más aconsejable sería de 2 a 5 segundos, en este último caso deberíamos multiplicarlo por 12

```
revo=revo * 12      ;multiplicar la variable revo por 12 para tener RPM
```

NOTA: Se puede utilizar los optoacopladores que vienen en los mouse de los PC, también es importante que el disco sea de color negro, para impedir que la luz infrarroja no pueda atravesar hacia el transistor, un disco de metal también podría funcionar bien.

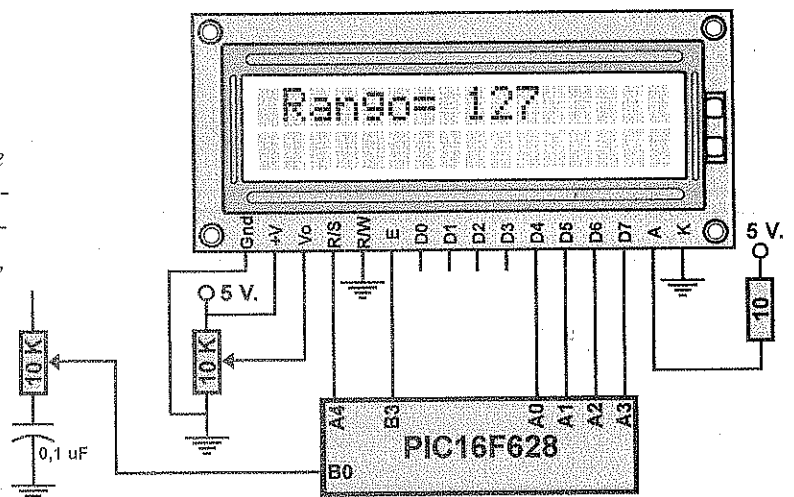
5.5.6. LECTURA DE UN POTENCIÓMETRO CON LCD.

LA DECLARACIÓN POT. Esta declaración sirve para leer el estado de un potenciómetro de 5K hasta uno de 50K, o cualquier elemento resistivo (fotoceldas, termistores, etc.), el principio de funcionamiento es muy sencillo utiliza la ayuda de un condensador de 0,1 uF, al cual lo carga y lo descarga utilizando el potenciómetro para regular la corriente que circula, entonces a mayor resistencia el capacitor se demora más tiempo en cargarse, y la variable nos dará un valor alto y si giramos el potenciómetro a la mínima resistencia, el capacitor se cargará más rápido y la variable nos dará cero, en definitiva estaremos leyendo el estado de un potenciómetro, el cual podemos aplicarlo en la atenuación de una luz por ejemplo o la variación de la velocidad de un motor.

MATERIALES.

- 1 DISPLAY LCD 2x16
- 1 resistencia de 10 Ω
- 2 potenciómetros de 10 K Ω
- 1 condensador cerámico de 0,1uF (referencia 104)

Figura 5.5.6.1. Conexión de un potenciómetro para leerlo.
En este caso el potenciómetro se encuentra en la mitad de su escala es decir 5 K Ω , esto equivale a 127, y si estuviera en 10K Ω , equivaldría a 255.



CMCON = 7	;convierte en digitales el puerto A
dato VAR BYTE	;variable dato con capacidad de 255
medir:	
POT portb.0,255,dato	;leer el potenciómetro y guardar en dato
LCDOUT \$FE, 1," Rango="	;limpiar pantalla y escribir rango=
LCDOUT #dato	;mostrar el valor decimal de dato
PAUSE 100	
GOTO medir	
END	

Figura 5.5.6.2. potenciometro.bas Programa para leer el estado de un potenciómetro.

Una vez terminado este proyecto, reemplace el potenciómetro con una fotocelda o un termistor.

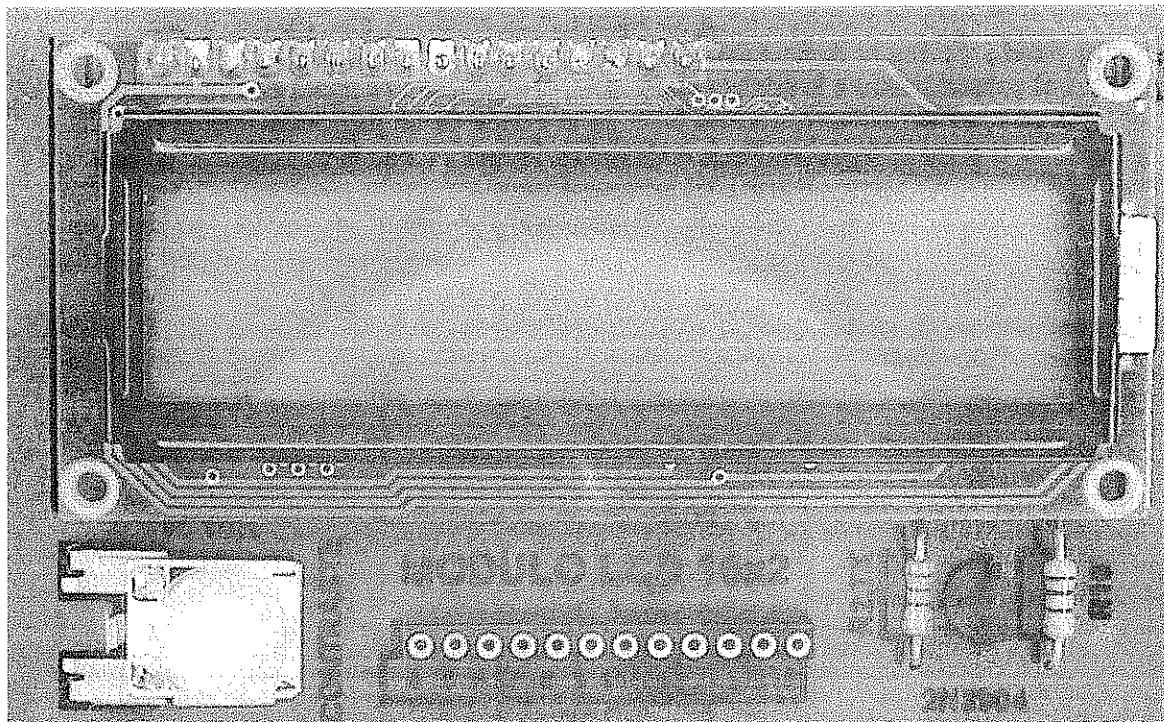


Figura 5.5.6.3. Fotografía del módulo LCD del entrenador experto de PIC'S EE-01 de AUTOMASIS.

5.6 SONIDO

5.6.1. GENERACIÓN DE SONIDO.

El compilador PBP es capaz de sacar las frecuencias especificadas por un pin del PIC, para esto aprenderemos la declaración **FREQOUT**.

LA DECLARACIÓN FREQOUT. Saca la o las frecuencias especificadas por un pin del micro, estas pueden ser de 0 a 32767 Hz, su utilización es de la siguiente manera:

```
FREQOUT portb.0, 2000, 1000
```

Quiere decir sacar una frecuencia de 1000 ciclos (1 KHz) durante 2 segundos por el puerto B.0

En esta práctica lo utilizaremos para generar un sonido a través de un piezoeléctrico (Buzzer pasivo) o podría ser también un parlante.

MATERIALES.

- 1 buzzer pasivo (piezoeléctrico) o un parlante grande o pequeño
- 1 capacitor de 10uF/25V.

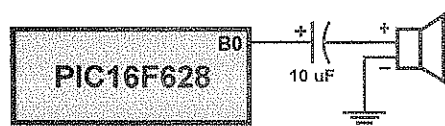


Figura 5.6.1.1. Diagrama de conexión de un parlante pequeño o un piezoeléctrico, este necesita de un capacitor para poder mejorar la señal del PIC.

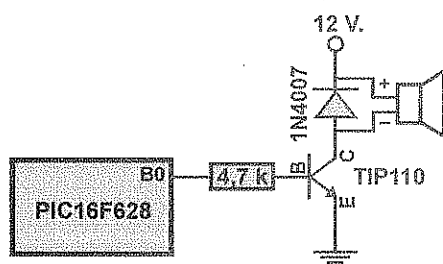


Figura 5.6.1.2. Diagrama de conexión para un parlante grande con un transistor TIP110.

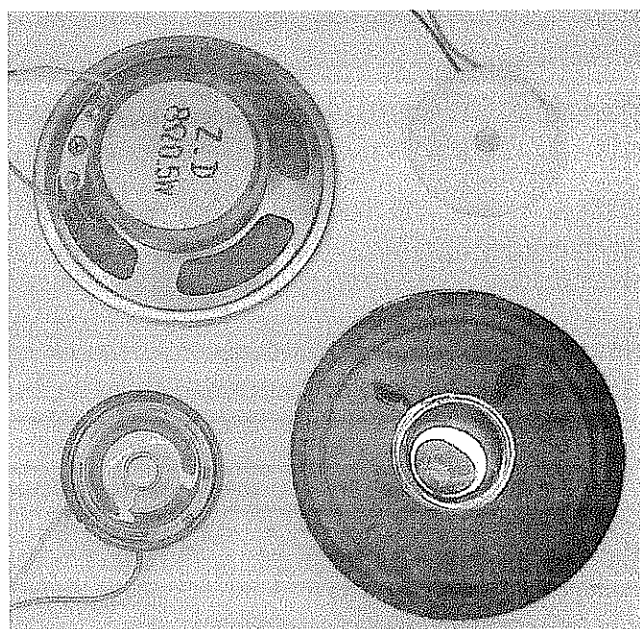


Figura 5.6.1.3. Fotografía de 2 parlantes grandes, un pequeño y un piezoeléctrico.

NOTA: también es posible utilizar los parlantes grandes con la conexión de la figura 5.6.1.1.

```

FREQOUT portb.0 , 2000 , 7200      ;sacar una frecuencia de 7,2 Khz
                                      ;durante 2 segundos por el Puerto B.0
END

```

Figura 5.6.1.4. freqout.bas Programa para generar sonido a través de un parlante.

5.6.2. UNA SIRENA POLICIAL.

Esta práctica consiste en sacar por un parlante el sonido característico de una sirena policial, para esto emplearemos la ayuda de la declaración **SOUND**.

LA DECLARACIÓN SOUND. Sirve para generar tonos y/o ruido blanco en un pin del PIC, y es posible combinar hasta 2 frecuencias desde de 1 a 127 que son tonos y 128 a 255 ruido blanco, 0 es silencio, 1 equivale a 78,74 HZ y 127 a 10000 Hz, esto se lo utiliza de la siguiente manera:

```
SOUND portB.0,[100,10,50,10]
```

Esto quiere decir sacar 2 tonos por el puerto b.0, el primer tono es 100 que equivale a (7874 Hz) con una duración de 10 milisegundos y luego un tono de 50 (3937 Hz) con una duración de 10 milisegundos también.

En cuanto a los materiales y diagrama de conexión son los mismos de la práctica anterior

```

Programa:
      SOUND portb.0 , [100,10,50,10]      ;genera tonos por el Puerto B.0
      GOTO programa
END

```

Figura 5.6.2.1. sirena.bas Programa para generar una sirena policial a través de un parlante.

UTILIZANDO UN CRISTAL DE MAYOR VELOCIDAD. Como sabemos el PIC esta trabajando actualmente a una velocidad de 4Mhz, utilizando un oscilador RC interno (resistencia condensador), pero el PIC puede operar también con osciladores externos de hasta 20 Mhz. Este es el momento de aprender a utilizar un oscilador de mayor frecuencia (8,10,12,16,20 Mhz), en este caso notaremos una considerable diferencia en cuanto a la nitidez del sonido respecto al programa 5.6.2.1, esto se logra adicionando un **DEFINE** al inicio del programa de la siguiente manera:

```
DEFINE OSC 20      ; especifica al PBP que se va a utilizar un cristal de 20 Mhz
```

Con un oscilador de 20Mhz, el micro trabaja 5 veces más rápido que antes es decir si con un oscilador interno de 4 Mhz, el PIC ejecutaba cada instrucción en 1 uS., con un oscilador de 20 Mhz lo hará en 0,2 uS (0,0000002 S).

Para esta práctica necesitamos estos nuevos elementos además de los anteriores:

MATERIALES.

- 1 cristal de 12 MHZ como el de la figura 5.6.2.2. ideal para protoboards
- 2 condensadores de 22 pF (22 picoFaradios)

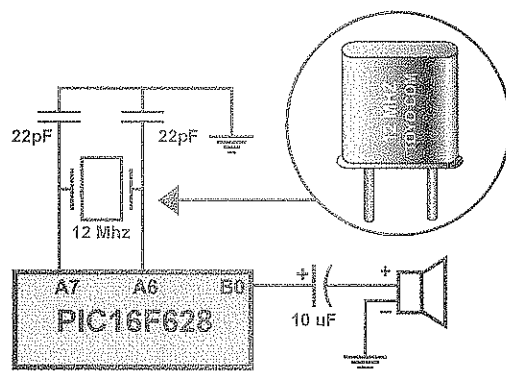


Figura 5.6.2.2. Diagrama de conexión de un cristal externo, muestra también la forma de este.
En el caso de utilizar un cristal de 16 Mhz, se debe poner : **DEFINE OSC 16**

```

DEFINE OSC 12                ;especifica que se va a utilizar un oscilador de 12 Mhz
Programa:
    SOUND portb.0 , [100,10,50,10]    ;genera tonos por el Puerto B.0
    GOTO programa
END

```

Figura 5.6.2.3. sirena12Mhz.bas Program para generar una sirena policial mejor que la anterior

IMPORTANTE: Este proyecto no funciona si no se especifica en el IC-prog que se va a utilizar un cristal externo de 12 Mhz, para esto debemos seleccionar HS en donde dice configuración Oscilador.
El PIC16F628, funciona muy bien con un cristal de 20 MHZ, lo que no sucede con el PIC16F628A, esta nueva serie se debe utilizar con cristales hasta de 16 MHZ.

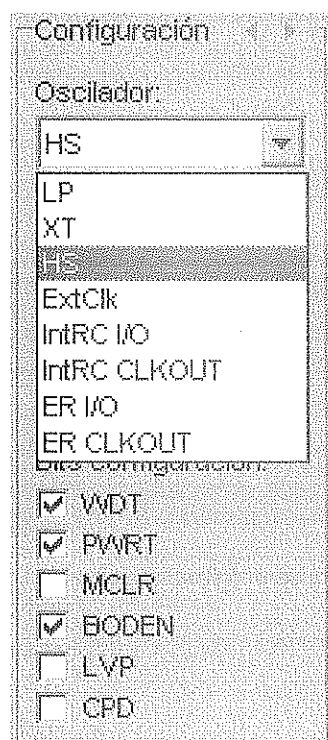


Figura 5.6.2.4. Configuración manual del Oscilador del programa IC-prog, para este ejercicio en el que se va a utilizar un cristal Externo de 12 MHZ, debemos seleccionar Oscilador HS

Frecuencia	OSC1/ C1	OSC2/ C2	Tipo
4 Mhz	-----	-----	IntRC I/O
32 KHz	68 – 100 pF	68 – 100 pF	LP
200 KHz	15 – 30 pF	15 – 30 pF	
100 KHz	68 – 150 pF	68 – 150 pF	XT
2 Mhz	15 – 30 pF	15 – 30 pF	
4 Mhz	15 – 30 pF	15 – 30 pF	
8 Mhz	15 – 30 pF	15 – 30 pF	HS
10 Mhz	15 – 30 pF	15 – 30 pF	
12 Mhz	15 – 30 pF	15 – 30 pF	
16 Mhz	15 – 30 pF	15 – 30 pF	
20 Mhz	15 – 30 pF	15 – 30 pF	

Figura 5.6.2.5. *Tabla de Configuración de Oscilador para el IC-prog, según el cristal a utilizar, además indica los valores de capacitores que se deben poner.*

También podemos agregar una línea de código ensamblador para que se encargue de cambiarnos el oscilador predefinido a HS, esto se consigue de la siguiente manera, debemos insertarlo al principio del programa lo siguiente:

```
@ device HS_OSC          ;cambia automáticamente a oscilador de alta velocidad HS
```

Para comprobarlo una vez compilado abra el archivo en el programa IC-prog y observará que el oscilador ha cambiado a HS sin que usted haga nada.

5.6.3. GENERACIÓN DE UN TIMBRE DE TELÉFONO CELULAR.

Esta práctica consiste en sacar por un parlante el sonido característico de un teléfono celular, para esto emplearemos la ayuda de la declaración **SOUND**.

En cuanto al diagrama de conexión y materiales, podemos utilizar el mismo de la práctica 5.6.1.

```
x VAR BYTE          ;variable x de 255
prog:
FOR X= 1 TO 15      ;repetir de 1 a 15 veces
  SOUND portB.0,[125,4,123,5] ;sacar tonos por el puerto b.0
NEXT                ;siguiente repetición
PAUSE 2500          ;esperar 2,5 segundos antes de volver a timbrar
GOTO prog
```

Figura 5.6.3.1. *timbre-celular.bas Programa para generar un timbre de teléfono celular.*

Pruebe con algunas combinaciones de tonos, así como también con diferentes tiempos y verá que bien puede componer una melodía.

5.6.4. LLAMADA TELEFÓNICA DTMF.

Esta práctica tiene como objetivo generar tonos DTMF (Dual-Tono MultiFrequency) a través de un PIC, como los que genera cualquier teléfono fijo o celular, estos tonos no son nada más que el envío de 2 frecuencias específicas asignadas a cada tecla, estas frecuencias podemos ver en la figura 5.6.4.1., lo que sucede cuando pulsamos una tecla por ejemplo la 6, es que este envía una frecuencia del grupo bajo 770 Hz y luego una segunda frecuencia del grupo alto 1477 Hz, esto podríamos hacerlo generando con la declaración **FREQOUT**.

FREQOUT portb.0,200,941,1336	;equivale a presionar la tecla 0 de DTMF, durante 200 mls
PAUSE 50	;retardo de 50 mls antes de pulsar la siguiente tecla
FREQOUT portb.0,200,852,1477	;equivale a presionar la tecla 9 de DTMF, durante 200 mls
PAUSE 50	;retardo de 50 mls antes de pulsar la siguiente tecla

Esto deberíamos hacer por cada tecla que deseáramos que marque, pero para facilitarnos las cosas el compilador PBP tiene una declaración específica para este trabajo.

LA DECLARACIÓN DTMFOUT. Esta genera automáticamente los tonos duales correspondientes a cada tecla y los envía cada una con intervalos de 50 milisegundos, aunque los tiempos podemos cambiarlo si lo deseamos (ver manual de pbp) su manera de utilizar es la siguiente:

DTMFOUT portb.0,[0,9,7,6,3,0,2,6,8]	; equivale a presionar las teclas 0,9,7,6,3,0,2,6,8
--	---

Grupo de frecuencias altas		1209Hz	1336Hz	1477Hz	1633Hz
Grupo de frecuencias bajas	679Hz	1	2 ABC	3 DEF	A
	770Hz	4 GHI	5 JKL	6 MNO	B
	852Hz	7 PQRS	8 TUV	9 WXYZ	C
	941Hz	*	0	#	D

Figura 5.6.4.1. Tabla de las frecuencias DTMF correspondiente a cada tecla, las teclas A,B,C,D, son para aplicaciones especiales, y no se los encuentra en los teclados comunes.

En la figura 5.6.4.2., se muestra el diagrama básico de conexión para poder hacer una llamada telefónica, debido a que el oscilador interno que posee el PIC es un RC (resistencia condensador), este no es muy preciso, por lo que experimentalmente se comprobó que el 70 % de los intentos para marcar los tonos DTMF llama a un número de teléfono incorrecto, y solo el 30% restante de intentos son los números deseados.

Para que los tonos DTMF que generan el PIC sean válidos al 100%, es necesario utilizar un cristal externo, sea este desde 4 a 20 MHz, con sus debidos capacitores.

La resistencia de 560Ω a 1 vatio paralela a la red telefónica, sirve para simular la carga de un teléfono normal, y con esto podremos tener el tono de marcado necesario para hacer la llamada, además notarán que esta resistencia empieza a disipar calor en el momento que se une a la red telefónica, esto debido a la cantidad de voltaje que circula (60 voltios).

El relé hace la conexión y desconexión de la red telefónica, que en este caso sería como el auricular que cuelga y descuelga el teléfono.

El filtro de $1\mu F$ a 100 voltios, sirve para poder mejorar la onda que sale del PIC y además como protección para el PIC. Es importante indicar que la red telefónica suministra alrededor de 60 voltios en DC, por lo que el capacitor debe pasar de 60 voltios y además el lado positivo de la red telefónica debe ir a tierra del PIC y el negativo hacia el pin del PIC, por lo que necesitaremos la ayuda de un voltímetro para poder identificar la polaridad de la red.

Para solucionar este inconveniente se propone el diagrama de la figura 5.6.4.3., este es un diagrama más completo en el que la polaridad de la línea no es un problema, puesto que dispone un puente de diodos en donde el lado positivo ya está unido a tierra y el lado negativo va a través del filtro hacia el PIC.

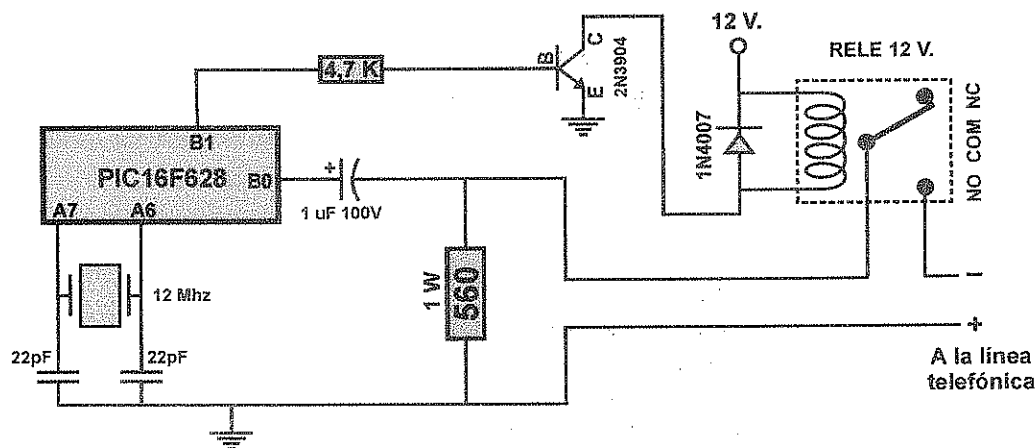


Figura 5.6.4.2. Diagrama básico de conexión para generar una llamada telefónica, noten que el lado positivo de la red telefónica debe ir a tierra del PIC, caso contrario no se podrá generar la llamada.

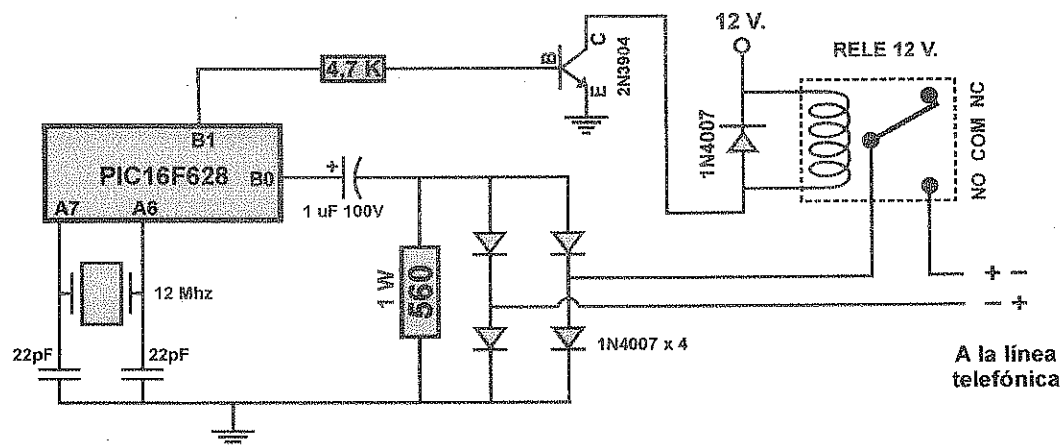


Figura 5.6.4.3. Diagrama de conexión para generar una llamada telefónica, en la que no importa la polaridad de la red telefónica, puesto que dispone un puente de diodos.

MATERIALES.

- 1 cristal de 12 MHZ, o cualquier otro de 4 a 16 MHZ
- 2 condensadores de 22 pF (22 picoFaradios)
- 1 capacitor de 1 uF a 100voltios
- 5 diodos rectificadores 1N4007
- 1 resistencia de 4,7 K Ω
- 1 resistencia de 560 Ω a 1 vatio
- 1 transistor 2N3904
- 1 relé a 12 voltios de 5 patitas

Este proyecto es muy utilizado en sistemas de seguridad, se puede hacer una alarma la cual en el momento de violar su seguridad, este realice automáticamente una llamada al propietario, el propietario en el momento de contestar escuchará una sirena, señal suficiente para saber que alguien ha activado la alarma, este mismo principio utilizan las centrales de monitoreo, con la diferencia que en vez de generar un sonido de sirena, se envía datos en forma serial como: Qué zona se abrió, a qué hora, estado de batería, etc. lo cual aprenderemos más adelante en comunicaciones con microcontroladores PIC.

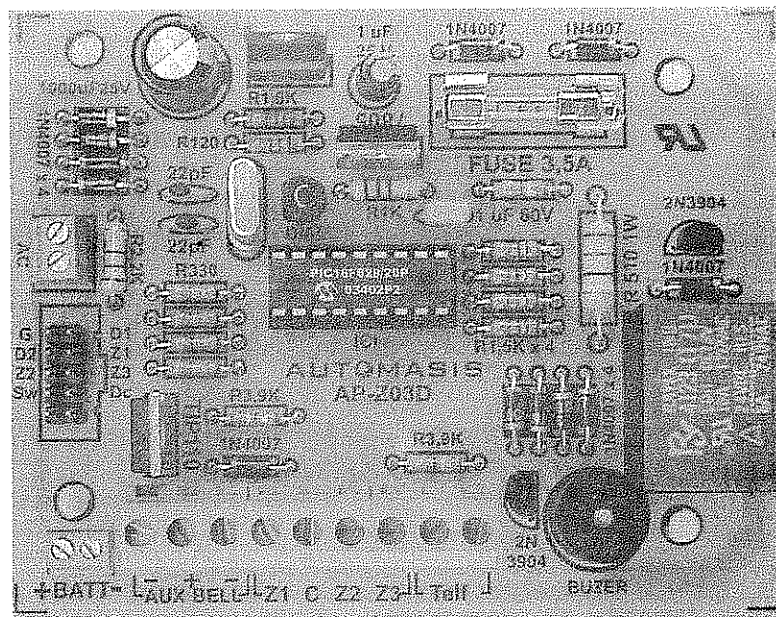


Figura 5.6.4.4. Fotografía de una tarjeta electrónica de una alarma de 3 zonas con armado de llave, cargador de batería y llamada telefónica, comandado por un PIC16F628.

```
@ device HS_OSC      ;cambia a oscilador HS en el icprog
DEFINE OSC 12        ;definir oscilador externo de 12 MHZ
relé VAR portb.1      ;nombre relé para el pin B.1
x VAR BYTE           ; variable x con tamaño de 255
```

iniciar:

continúa ...

PAUSE 2000	;espera de 2 segundos antes de empezar
HIGH rele	;conecta a la línea telefónica
PAUSE 1000	;espera 1 segundo hasta que exista tono de marcar
DTMFOUT portb.0,[0,9,7,6,3,0,2,6,8]	;número al cual el PIC va a llamar
PAUSE 3000	;esperar 3 seg. hasta que alguien conteste
FOR x = 1 TO 25	;repetir 25 veces , equivale a 6 segundos
SOUND portb.0,[100,10,50,10]	;enviar el sonido de sirena por el teléfono
NEXT	
LOW rele	;desconecta el relé , el cual cierra la llamada
END	

Figura 5.6.4.5. Llamada DTMF.bas Programa para hacer una llamada al teléfono celular.

Pruebe sus conocimientos intente hacer una alarma para su casa como el del siguiente gráfico:

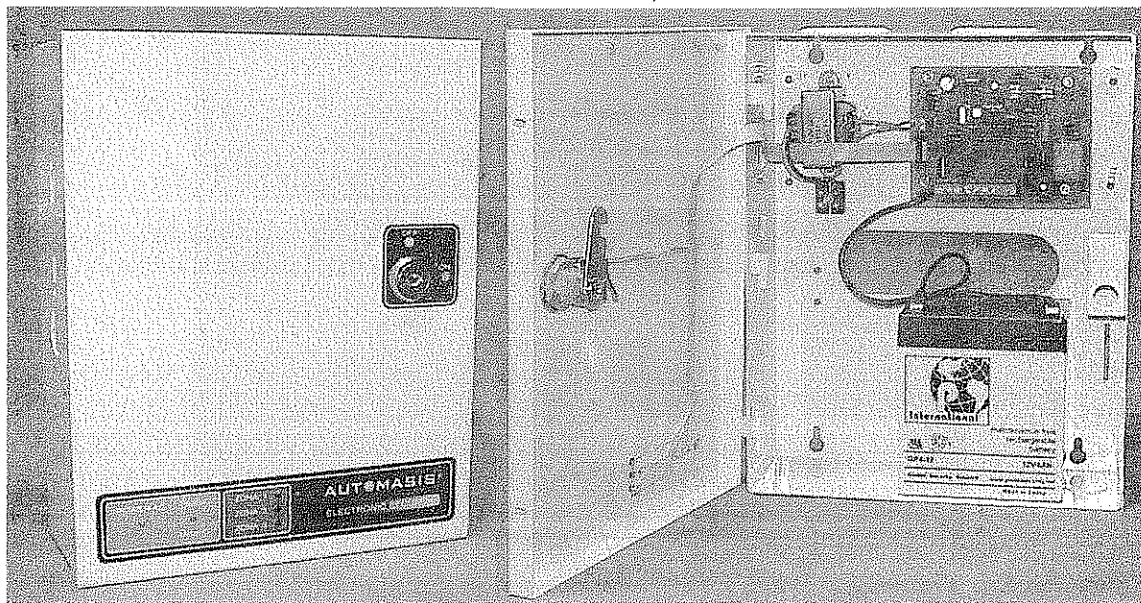


Figura 5.6.4.6. Fotografía de la alarma de 3 zonas basado en un PIC16F628, dispone de 3 leds para indicar la zona que está abierta, un led que indica si está armado o desarmado y un led que indica si hay poder AC, además del cargador de batería y llamada telefónica.

5.7 PROYECTOS CON TECLADOS

5.7.1. LECTURA DE UN TECLADO DE 16 PULSADORES CON DISPLAY DE 7 SEGMENTOS.

Los teclados matriciales son muy útiles para ingresar datos, un ejemplo es el teclado del computador, el teclado de una alarma que nos permite armar y desarmar un sistema de seguridad, el teclado de una caja fuerte, el de una cerradura eléctrica, etc. Para introducirnos en el manejo de un teclado, haremos un proyecto para aprender a identificar filas y columnas que lo componen un teclado hexadecimal de 16 pulsadores, y su correspondiente barrido de teclas, luego se visualizará en un display de 7 segmentos el número de la tecla presionada.

MATERIALES.

- 1 teclado matricial hexadecimal de 16 teclas como el de la figura 5.7.2.3
- 7 resistencias de 330Ω
- 4 resistencia de $4,7\text{ K}\Omega$
- 1 DISPLAY de 7 segmentos ánodo común
- 1 CI. 7447 decodificador BCD.

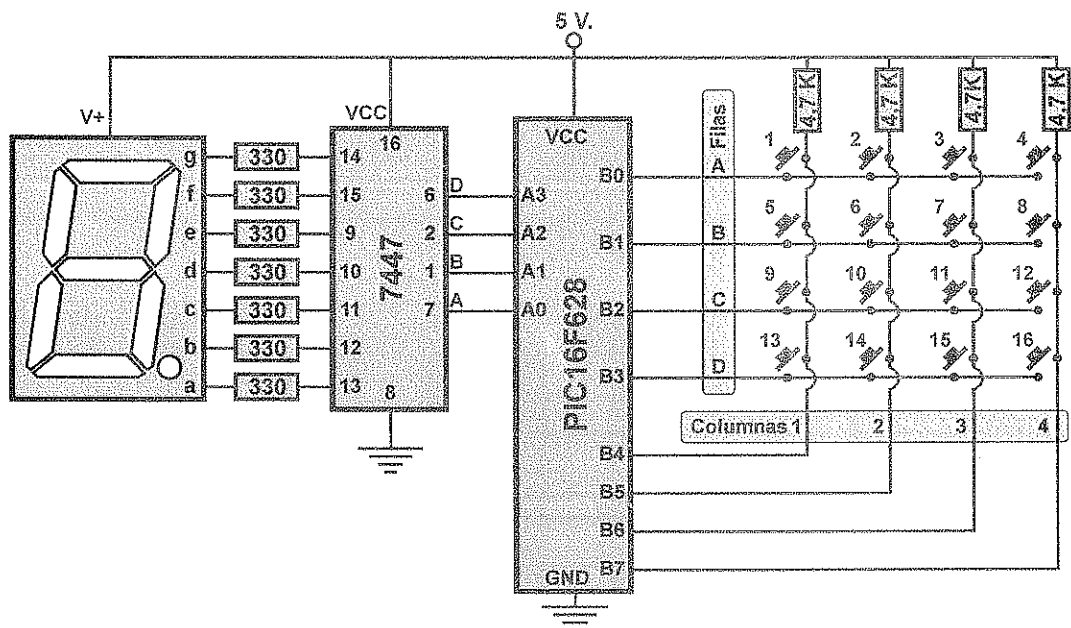


Figura 5.7.1.1. Diagrama de conexión de un teclado hexadecimal y un display de 7 segmentos.

CMCON=7	;convierte en pines digitales el puerto A	
fila VAR BYTE	;variable para las filas	
colu VAR BYTE	;variable para las columnas	
tecla VAR BYTE	;variable para almacenar el número de la tecla	
TRISA=0	;todo el puerto A configurado como salidas	continúa ...


```

prog1:
PORTB=0                                ;el puerto B es = %00000000
TRISB=%11110000                       ;configura 4 pines bajos como salida y los demás entrada
IF ((PORTB >> 4) != %1111) THEN prog1  ;si la tecla es presionada manténgalo en prog1
prog2:
FOR fila= 0 TO 3                       ;repetir para las 4 filas del teclado
PORTB=0                                ;el puerto B es = %00000000
TRISB=(DCD fila)^%11111111            ;setea una fila a 1 y los invierte a todos los demás
colu= PORTB >> 4                       ;desplaza los 4 bits altos al inicio
IF colu != %1111 THEN numtecla        ;si una tecla es pulsada ir numtecla
NEXT fila
GOTO prog2

numtecla:
tecla = (fila*4)+(NCD (colu^%1111))    ;calcula el valor de la tecla multiplicando
                                         ;por 4 la fila a la que pertenece y sumando a la posición que se encuentra
                                         ;para un teclado de 12 pulsadores , debemos cambiar (fila*3)
porta=tecla                           ;sacar por el puerto A el valor de tecla
GOTO prog1
END

```

Figura 5.7.1.2. teclado16-display.bas Programa para leer un teclado hexadecimal y mostrarlo en un display de 7 segmentos.

Debe considerarse que el lugar de las teclas no se pueden cambiar, puesto que este sistema de programación, utiliza operaciones matemáticas para calcular la tecla pulsada, pero debemos reconocer lo pequeño que es el programa, por eso se propone otro modo de programar, en donde los valores se le puede asignar en cualquier lugar, así como también se le puede poner letras.

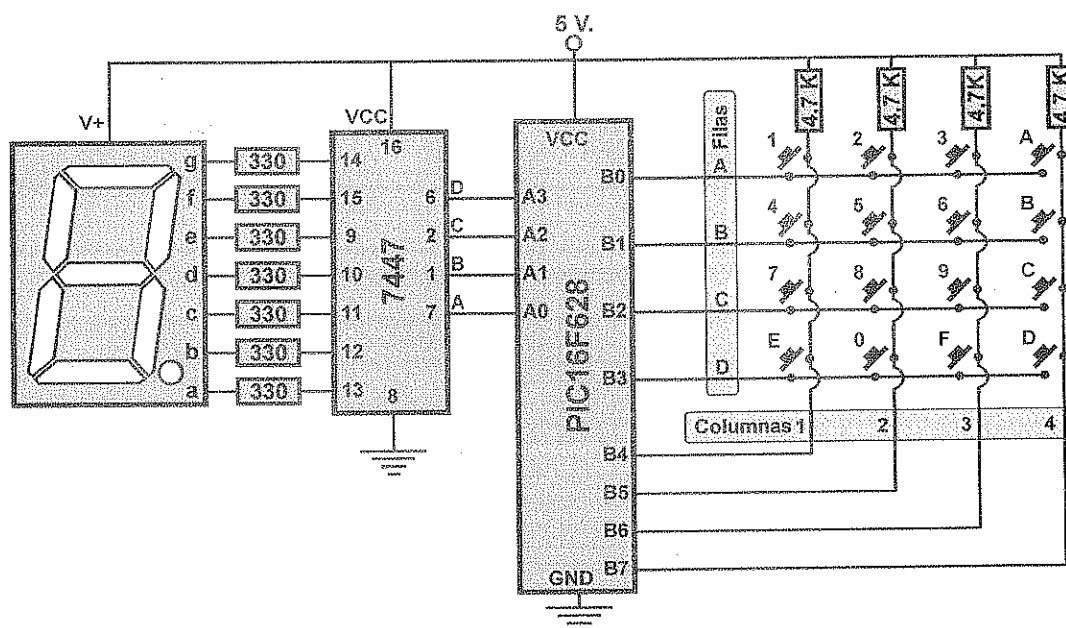


Figura 5.7.1.3. Diagrama de conexión de un teclado hexadecimal con diferente lugar de teclas.

cmcon=7	;cambiar a modo digital todo el puerto A
TRISA = 0	;todo el puerto A es configurado como salidas
A VAR PORTB.0	;nombres para los pines de las filas
B VAR PORTB.1	
C VAR PORTB.2	
D VAR PORTB.3	
UNO VAR PORTB.4	;nombres para los pines de las columnas
DOS VAR PORTB.5	
TRES VAR PORTB.6	
CUATRO VAR PORTB.7	
BARRIDO:	
LOW A	;hacer bajo la 1era fila
IF UNO = 0 THEN PORTA = 1	;si la 1ra tecla es presionada sacar 1
IF DOS = 0 THEN PORTA = 2	;si la 2da tecla es presionada sacar 2
IF TRES = 0 THEN PORTA = 3	;si la 3ra tecla es presionada sacar 3
IF CUATRO = 0 THEN PORTA = 10	;si la 4ta tecla es presionada sacar 10
HIGH A	;poner en alto la 1era fila
LOW B	;hacer bajo la 2da fila
IF UNO = 0 THEN PORTA = 4	;si la 1ra tecla es presionada sacar 4
IF DOS = 0 THEN PORTA = 5	; y así sucesivamente
IF TRES = 0 THEN PORTA = 6	
IF CUATRO = 0 THEN PORTA = 11	
HIGH B	;poner en alto la 2da fila
LOW C	;hacer bajo la 3ra fila
IF UNO = 0 THEN PORTA = 7	
IF DOS = 0 THEN PORTA = 8	
IF TRES = 0 THEN PORTA = 9	
IF CUATRO = 0 THEN PORTA = 12	
HIGH C	;poner en alto la 3ra fila
LOW D	;hacer bajo la 4ta fila
IF UNO = 0 THEN PORTA = 14	
IF DOS = 0 THEN PORTA = 0	
IF TRES = 0 THEN PORTA = 15	
IF CUATRO = 0 THEN PORTA = 13	
HIGH D	;poner en alto la 4ta fila
PAUSE 10	;pausa de 10 milisegundos
GOTO BARRIDO	;continuar con el barrido de teclas
END	

Figura 5.7.1.4. teclado16-display2.bas Programa diferente para leer un teclado hexadecimal y mostrarlo en un display de 7 segmentos.

Como se puede ver este programa es un poco más largo, pero ocupa menos espacio en la memoria del PIC que el ejercicio anterior, además tiene la ventaja de poder poner en cualquier lugar el valor de las teclas y es más fácil de entender, así que este será la forma que utilizaremos en adelante. Su funcionamiento es sencillo solo debemos fijarnos cual fila es la que está en **LOW** y

esta es la fila que se está barriendo, si una de las condiciones encuentra la igualdad, pues esta es la tecla pulsada.

Ejemplo: si pulsamos la tecla 6, en algún momento se pondrá en bajo la fila B y detectará un cambio de estado de 1 a 0 en la columna 3 (puerto B.6), por lo que:

```
LOW B
IF TRES=0 THEN PORTA = 6      ; detecta un cambio de estado y saca por el display el # 6
HIGH B
```

Debemos considerar que una persona requiere como mínimo 100 milisegundos para presionar una tecla, en ese tiempo el PIC realiza 10 barridos, por lo que de seguro detectará inmediatamente la tecla pulsada.

Para el caso de utilizar un teclado de 12 pulsadores, debemos eliminar una columna, las que corresponde a la tecla A, B , C, y D, es decir eliminamos las siguientes líneas del programa:

```
IF CUATRO = 0 THEN PORTA= 10
IF CUATRO = 0 THEN PORTA= 11
IF CUATRO = 0 THEN PORTA= 12
IF CUATRO = 0 THEN PORTA= 13
```

En este caso quedaría un teclado con pulsadores del 0 al 9 y dos teclas de propósito especial, la tecla asterisco (*) y la tecla numeral (#).

5.7.2. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA FLASH

Este es un proyecto aplicable en seguridad, se trata de una cerradura electrónica en la cual al ingresar los 4 dígitos correctamente en su teclado, el PIC energiza un relé (puerta), pero si la clave es incorrecta el PIC emite 3 pitos indicando que ingresó una clave errónea y por supuesto que el relé no se conectará, para hacerlo más interesante se le ha agregado sonido a las teclas en el momento de ser pulsadas esto sirve para que el usuario sepa que el PIC reconoció la pulsación, también tiene un programa antirrebote de tecla para asegurarse que ingrese una sola tecla a la vez.

El único inconveniente es que la clave no puede ser cambiada, ya que el número de la combinación (1,2,3,4) se lo grabó en el programa del PIC por lo que reside en la memoria FLASH, más adelante en los próximos proyectos la clave podrá ser cambiada a gusto del usuario, en tal caso este proyecto sirve para poder identificar y diferenciar las 3 memorias que dispone el PIC.

MATERIALES.

- 1 teclado matricial hexadecimal como el de la figura 5.7.2.3
- 6 resistencia de 4,7 K Ω
- 1 resistencia de 330 Ω
- 2 transistores 2N3904
- 1 chicharra activa (las que suenan directamente al alimentarles con 12 voltios)
- 1 relé de 12 voltios de 5 patitas
- 1 LED rojo de 5 mm.
- 1 Diodo rectificador 1N4007.

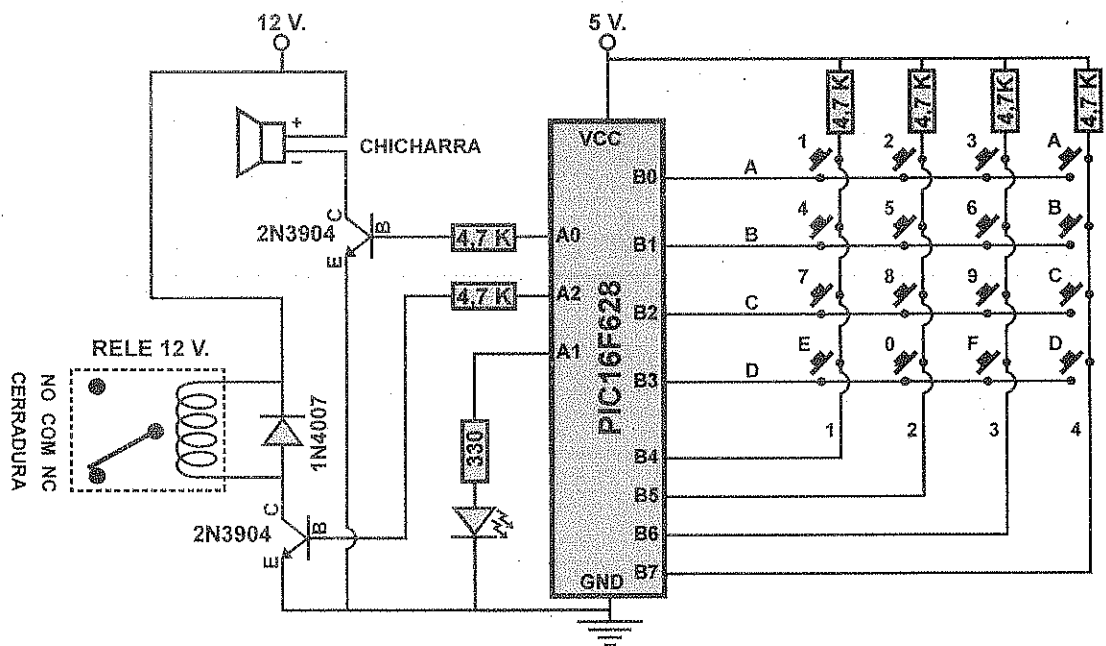


Figura 5.7.2.1. Diagrama de conexión de un teclado hexadecimal con un relé un led y una chicharra, para hacer una cerradura electrónica.

```

cmcon=7                                ;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE                         ;variable número para almacenar la tecla pulsada
R VAR BYTE                             ;variable r para hacer repeticiones

BIP VAR PORTA.0                        ;el portA.1 Bip (conectar chicharra activa)
LED VAR PORTA.1                        ;el portA.2 se llamará led
DOOR VAR PORTA.2                       ;el portA.3 conectar relé para la cerradura

A VAR PORTB.0                          ;nombres para los pines de las filas
B VAR PORTB.1
C VAR PORTB.2
D VAR PORTB.3

UNO  VAR PORTB.4                       ;nombres para los pines de las columnas
DOS  VAR PORTB.5
TRES VAR PORTB.6
CUATRO VAR PORTB.7

INICIANDO:                             ;programa del led para saber si está funcionando
    HIGH LED : HIGH BIP
    PAUSE 500
    LOW LED : LOW BIP

GOTO TECLAUNO                          ;ir a comparar claves

BARRIDO:
    LOW A                               ;sensar la fila A

```

continúa


```

IF UNO = 0 THEN NUMERO = 1 : RETURN ;tecla retorna cargada con 1
IF DOS = 0 THEN NUMERO = 2 : RETURN ;tecla retorna cargada con 2
IF TRES = 0 THEN NUMERO = 3 : RETURN ;tecla retorna cargada con 3
IF CUATRO = 0 THEN NUMERO = 10 : RETURN ;tecla retorna cargada con 10
HIGH A
LOW B ;sensar la fila B
IF UNO = 0 THEN NUMERO = 4 : RETURN
IF DOS = 0 THEN NUMERO = 5 : RETURN
IF TRES = 0 THEN NUMERO = 6 : RETURN
IF CUATRO = 0 THEN NUMERO = 11 : RETURN
HIGH B
LOW C ;sensar la fila C
IF UNO = 0 THEN NUMERO = 7 : RETURN
IF DOS = 0 THEN NUMERO = 8 : RETURN
IF TRES = 0 THEN NUMERO = 9 : RETURN
IF CUATRO = 0 THEN NUMERO = 12 : RETURN
HIGH C
LOW D ;sensar la fila D
IF UNO = 0 THEN NUMERO = 14 : RETURN
IF DOS = 0 THEN NUMERO = 0 : RETURN
IF TRES = 0 THEN NUMERO = 15 : RETURN
IF CUATRO = 0 THEN NUMERO = 13 : RETURN
HIGH D
PAUSE 10
GOTO BARRIDO

```

***** programa de antirrebote de teclas *****

PTECLA:

```

HIGH LED : HIGH BIP ;genera sonido cada que se pulsa tecla
PAUSE 100 ;duración 100 milisegundos
LOW LED : LOW BIP ;apagar sonido y led
ESPACIO: ;programa de antirrebote de teclas
IF UNO = 0 THEN ESPACIO ;si la tecla sigue pulsada ir a espacio
IF DOS = 0 THEN ESPACIO ;si la tecla sigue pulsada ir a espacio
IF TRES = 0 THEN ESPACIO ;si la tecla sigue pulsada ir a espacio
IF CUATRO = 0 THEN ESPACIO ;si la tecla sigue pulsada ir a espacio
PAUSE 25
RETURN ;retorna si se suelta las teclas

```

***** comparación de claves *****

TECLAUNO:

```

GOSUB BARRIDO ;ir a barrido y retornar con un valor
GOSUB PTECLA ;envía a un programa antirrebote para soltar tecla
IF NUMERO = 1 THEN TECLADOS ;si el número es igual a 1 ir teclados
GOTO FALSO ;caso contrario ir a lazo falso

```

TECLADOS:

```

GOSUB BARRIDO: GOSUB PTECLA ;ir a barrido y retornar con un valor

```

Continúa ...


```

IF NUMERO = 2 THEN TECLATRES           ;si el número es igual a 2 ir teclatres
    GOTO FALSO1                         ;caso contrario ir a lazo falso

TECLATRES:
    GOSUB BARRIDO :GOSUB PTECLA         ;ir a barrido y retornar con un valor
IF NUMERO = 3 THEN TECLACUATRO         ;si el número es igual a 3 ir teclacuatro
    GOTO FALSO2                         ;caso contrario ir a lazo falso

TECLACUATRO:
    GOSUB BARRIDO :GOSUB PTECLA         ;ir a barrido y retornar con un valor
IF NUMERO = 4 THEN OPENGE               ;si el número es igual a 4 conectar relé
    GOTO FALSO3                         ;caso contrario ir a lazo falso

OPENGE:
    FOR R = 1 TO 2                     ;2 pitos indica clave correcta
        PAUSE 100
        HIGH LED : HIGH BIP
        PAUSE 100
        LOW LED : LOW BIP
    NEXT

    HIGH DOOR                           ; se conecta el relé (abrir puerta)
    PAUSE 1000                           ; esperar 1 segundo
    LOW DOOR                             ; desconectar relé
    GOTO TECLAUNO                        ;ir nuevamente a comparar las claves

; ***** lazos falsos teclas erroneas *****
FALSO:
    GOSUB BARRIDO :GOSUB PTECLA         ;estas teclas no comparan ninguna
FALSO1:
    GOSUB BARRIDO :GOSUB PTECLA         ;clave solo espera que termine de
FALSO2:
    GOSUB BARRIDO :GOSUB PTECLA         ;pulsar las 4 teclas y no hace nada

FALSO3:
    FOR R = 1 TO 3                     ;3 pitos indica clave incorrecta
        PAUSE 100
        HIGH LED : HIGH BIP
        PAUSE 100
        LOW LED : LOW BIP
    NEXT
    GOTO TECLAUNO                       ;ir nuevamente a comparar las claves
END

```

Figura 5.7.2.2. cerradura FLASH.bas Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave no se puede cambiar.

NOTA: Este experimento funcionó muy bien en el entrenador y en el protoboard, pero al armarlo en un circuito impreso, dejó de funcionar la chicharra, esto se soluciona adicionando un capacitor de 1 uF. en paralelo al PIC.

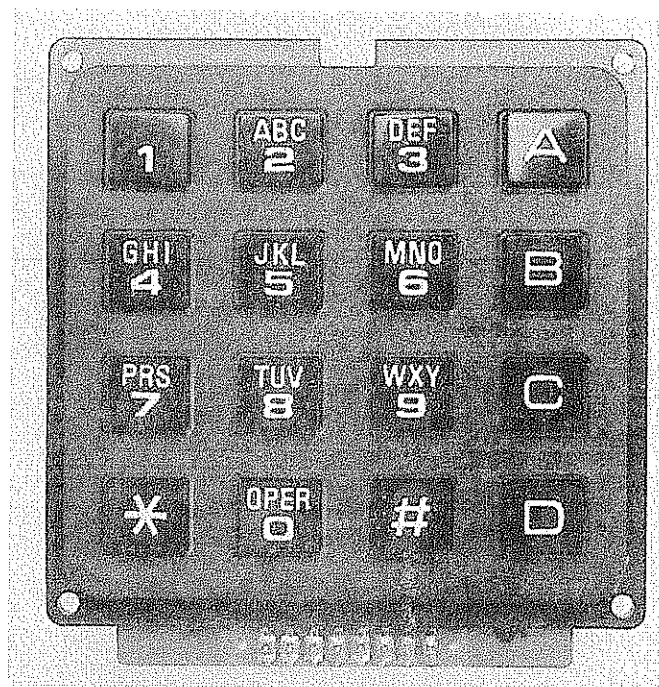


Figura 5.7.2.3. Fotografía de un teclado matricial hexadecimal común en las tiendas electrónicas.

5.7.3. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA RAM Y CAMBIO DE CLAVE.

Este proyecto es muy similar al anterior con la diferencia que este se le puede cambiar la clave predefinida (1,2,3,4) por cualquier otra combinación de teclas, la clave original será cargada en cuatro variables y existirá una manera de cambiar los valores de estas variables lo cual se lo hace de la siguiente manera: después de haber colocado la clave original (1,2,3,4), debemos mantener presionado la tecla D durante 2 segundos, para ser más exactos en el momento que el relé se conecta después de 1 segundo hay una pregunta de si la tecla D es presionada ir a grabar, si no presionamos la tecla D a tiempo, perderemos la oportunidad de cambiar la clave y tendremos que volver a repetir el proceso, en el momento que ingresa al programa de cambio de clave se encenderá el LED y permanecerá encendido esperando a que ingresemos los 4 nuevos dígitos.

Es importante saber que la nueva clave se almacenará en las variables SETPRIME, SETSEGUN, SETERCER, y SETCUART, estas variables ocupan espacio en la memoria RAM (Random Access Memory) o memoria de acceso casual que tiene una capacidad de 224 BYTES, por lo tanto solo están activas mientras el PIC se encuentra alimentado, una vez que se corta la alimentación al PIC esta memoria se borra (volátil), por consiguiente al momento de volver a prender el micro PIC, la nueva clave se nos habrá perdido y en su lugar se encontrará la clave original (1,2,3,4), esto debido a que esta clave se encuentra en la memoria FLASH y en el momento de correr el programa lo carga nuevamente en la memoria RAM para desde aquí poder ser modificada.

En cuanto a los materiales y el diagrama de conexión, son los mismos que se utilizaron en el ejercicio anterior.

cmcon=7	;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE	;variable número para almacenar la tecla pulsada
R VAR BYTE	;variable r para hacer repeticiones
BIP VAR PORTA.0	;el portA.1 Bip (conectar chicharra activa)
LED VAR PORTA.1	;el portA.2 se llamará led
DOOR VAR PORTA.2	;el portA.3 conectar relé para la cerradura
A VAR PORTB.0	;nombres para los pines de las filas
B VAR PORTB.1	
C VAR PORTB.2	
D VAR PORTB.3	
UNO VAR PORTB.4	;nombres para los pines de las columnas
DOS VAR PORTB.5	
TRES VAR PORTB.6	
CUATRO VAR PORTB.7	
SETPRIME VAR BYTE	;variable para almacenar la 1era clave
SETSEGUN VAR BYTE	;variable para almacenar la 2da clave
SETERCER VAR BYTE	;variable para almacenar la 3era clave
SETCUART VAR BYTE	;variable para almacenar la 4ta clave
SETPRIME =1	;variable cargada con la 1era clave
SETSEGUN =2	;variable cargada con la 2da clave
SETERCER =3	;variable cargada con la 3era clave
SETCUART =4	;variable cargada con la 4ta clave
INICIANDO:	;programa del led para saber si está funcionando
HIGH LED : HIGH BIP	
PAUSE 500	
LOW LED : LOW BIP	
GOTO TECLAUNO	;ir a comparar claves
GRABAUNO:	;programa para cambiar la clave
GOSUB PTECLA : HIGH LED	;espera a que suelte las teclas
GOSUB BARRIDO : GOSUB PTECLA	;ir a barrido y retorna a un antirrebote
HIGH LED	;mantener encendido el LED
SETPRIME = NUMERO	;guardar en setprime el valor de número
GRABADOS:	
GOSUB BARRIDO : GOSUB PTECLA	;ir a barrido y retorna a un antirrebote
HIGH LED	;mantener encendido el LED
SETSEGUN = NUMERO	;guardar el valor de número
GRABATRES:	
GOSUB BARRIDO : GOSUB PTECLA	;ir a barrido y retorna a un antirrebote
HIGH LED	;mantener encendido el LED
SETERCER = NUMERO	;guardar el valor de número

continúa

GRABACUATRO:

GOSUB BARRIDO : GOSUB PTECLA

HIGH LED

SETCUART = NUMERO

GOTO iniciando

;ir a barrido y retorna a un antirrebote

;mantener encendido el LED

;guardar el valor de numero

; ir a iniciando

BARRIDO:

LOW A

IF UNO = 0 THEN NUMERO =1 : RETURN

IF DOS = 0 THEN NUMERO =2 : RETURN

IF TRES = 0 THEN NUMERO =3 : RETURN

IF CUATRO=0 THEN NUMERO =10: RETURN

;sensar la fila A

;tecla pulsada retorne cargada con 1

;tecla pulsada retorne cargada con 2

;tecla pulsada retorne cargada con 3

;tecla pulsada retorne cargada con 10

HIGH A

LOW B

IF UNO = 0 THEN NUMERO =4 : RETURN

IF DOS = 0 THEN NUMERO =5 : RETURN

IF TRES = 0 THEN NUMERO =6 : RETURN

IF CUATRO=0 THEN NUMERO =11: RETURN

;sensar la fila B

HIGH B

LOW C

IF UNO = 0 THEN NUMERO =7 : RETURN

IF DOS = 0 THEN NUMERO =8 : RETURN

IF TRES = 0 THEN NUMERO =9 : RETURN

IF CUATRO=0 THEN NUMERO =12: RETURN

;sensar la fila C

HIGH C

LOW D

IF UNO = 0 THEN NUMERO =14: RETURN

IF DOS = 0 THEN NUMERO =0 : RETURN

IF TRES = 0 THEN NUMERO =15: RETURN

IF CUATRO= 0 THEN NUMERO =13: RETURN

;sensar la fila D

HIGH D

PAUSE 10

GOTO BARRIDO

; ***** programa de antirrebote de teclas *****

PTECLA:

HIGH LED : HIGH BIP

PAUSE 100

LOW LED : LOW BIP

ESPACIO:

IF UNO = 0 THEN ESPACIO

IF DOS = 0 THEN ESPACIO

IF TRES = 0 THEN ESPACIO

IF CUATRO= 0 THEN ESPACIO

PAUSE 25

RETURN

;genera sonido cada que se pulsa tecla

;duración 100 milisegundos

;apagar sonido y led

;programa de antirrebote de teclas

;si la tecla sigue pulsada ir espacio

;si la tecla sigue pulsada ir espacio

;si la tecla sigue pulsada ir espacio

;si la tecla sigue pulsada ir espacio

;retorna si se suelta las teclas

; ***** comparación de claves *****

TECLAUNO:

GOSUB BARRIDO

;ir a barrido y retornar con un valor

continúa ...

```

GOSUB PTECLA ;envía a un programa antirrebote para soltar tecla
IF numero = setprime THEN TECLADOS ;si el número es igual a setprime
GOTO FALSO ;caso contrario ir a lazo falso
TECLADOS:
GOSUB BARRIDO :GOSUB PTECLA ;ir a barrido y retornar con un valor
IF numero = setsegun THEN TECLATRES ;si el número es igual a setsegun
GOTO FALSO1 ;caso contrario ir a lazo falso
TECLATRES:
GOSUB BARRIDO :GOSUB PTECLA ;ir a barrido y retornar con un valor
IF numero = setercer THEN TECLACUATRO ;si el número es igual a setercer
GOTO FALSO2 ;caso contrario ir a lazo falso
TECLACUATRO:
GOSUB BARRIDO :GOSUB PTECLA ;ir a barrido y retornar con un valor
IF numero = setcuart THEN OPENGE ;si número es igual a setcuart conectar relé
GOTO FALSO3 ;caso contrario ir a lazo falso

OPENGE:
FOR R = 1 TO 2 ;2 pitos indica clave correcta
PAUSE 100
HIGH LED : HIGH BIP
PAUSE 100
LOW LED : LOW BIP
NEXT

HIGH DOOR ;se conecta el relé (abrir puerta)
PAUSE 1000 ;esperar 1 segundo
LOW DOOR ;desconectar relé
HIGH A : HIGH B : HIGH C : LOW D ;senar solo la fila D
IF CUATRO = 0 THEN GRABAUNO ;corresponde tecla D grabar
GOTO TECLAUNO ;ir nuevamente a comparar las claves

; ***** lazos falsos teclas erroneas *****
FALSO:
GOSUB BARRIDO :GOSUB PTECLA ;estas teclas no comparan ninguna
FALSO1: ;clave solo espera que termine de
GOSUB BARRIDO :GOSUB PTECLA ;pulsar las 4 teclas y no hace nada
FALSO2:
GOSUB BARRIDO :GOSUB PTECLA
FALSO3:
FOR R = 1 TO 3 ;3 pitos indica clave incorrecta
PAUSE 100
HIGH LED : HIGH BIP
PAUSE 100
LOW LED : LOW BIP
NEXT
GOTO TECLAUNO ;ir nuevamente a comparar las claves
END

```

Figura 5.7.3.1. cerradura_RAM.bas Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave se puede cambiar en la memoria RAM.

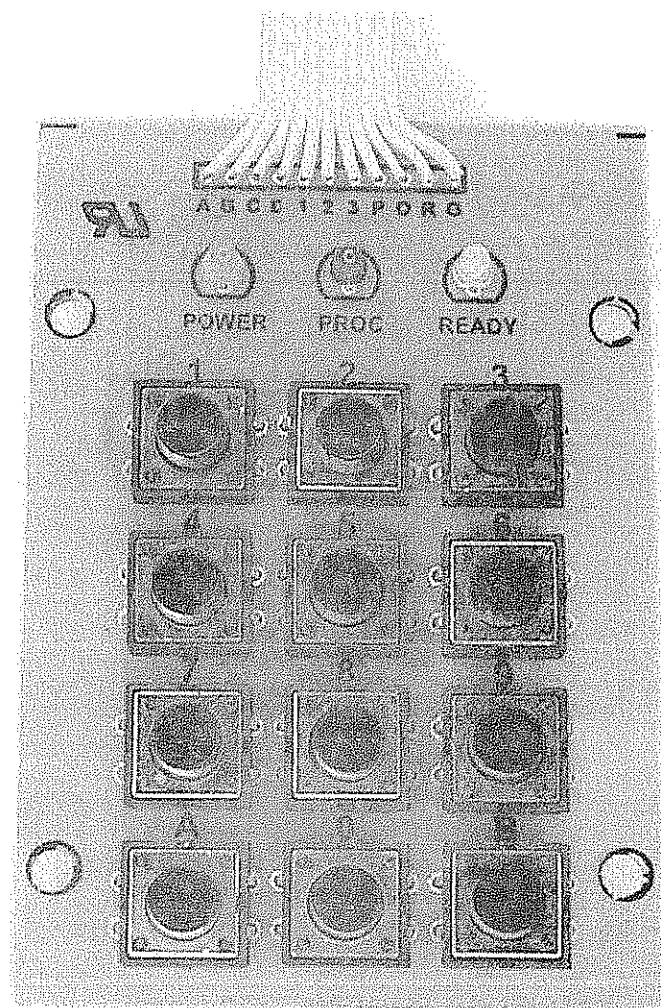




Figura 5.7.3.2. Fotografía de un teclado de 12 pulsadores y 3 leds indicadores, fabricado en una placa de fibra de vidrio.

5.7.4. CERRADURA ELECTRÓNICA CON CLAVE EN MEMORIA EEPROM Y CAMBIO DE CLAVE.

Una vez aprendido acerca de las 2 memorias anteriores, es el momento de aprender a utilizar la memoria EEPROM (Electrical Erasable Programable Read Only Memory), o memoria de lectura de programación y borrado eléctrico, que a diferencia de la memoria RAM, esta no es volátil y tiene capacidad para 128 Bytes, lo que quiere decir que si al PIC se le corta la alimentación, los datos almacenados en la memoria EEPROM, permanecen inalterados según su fabricante por un período de 40 años (PIC16F628) y 100 años (PIC16F628A).

Este proyecto cumple todas las expectativas de un estudiante de micros, tiene la posibilidad de cambiar la clave y no borrarse, si se digita una clave errónea el teclado se bloquea y únicamente *lo desbloqueamos presionando al mismo tiempo las teclas 7 y C por 2 segundos*, así mismo para cambiar la clave procedemos como en el proyecto anterior, presionando la *tecla D*.

Algo importante es que si nos olvidamos la clave, la única manera de recuperarlo es leyendo el contenido del PIC a través de programador IC-prog, esto se lo realiza colocando al PIC nuevamente en el grabador y presionando la tecla , la cual después de leer el contenido del PIC, nos mostrará la nueva clave almacenada  en la parte inferior, donde dice Dirección-Datos Eeprom.

En cuanto a los materiales y el diagrama de conexión, son los mismos que se utilizaron en el proyecto anterior.

LA DECLARACIÓN EEPROM, READ Y WRITE. Estas declaraciones las explicaremos con un ejercicio:

EEPROM 5,[3,"K",9,12] ; quiere decir colocar en la memoria EEPROM, dirección 5 el número 3, en la dirección 6 el carácter ASCII de K, es decir el número 75, aunque en el programa IC-prog lo veremos como 4B, esto porque está en sistema hexadecimal, continuando en la dirección 7 se guardará el número 9 y así sucesivamente, recuerden que el PIC16F628 tiene 128 espacios de memoria EEPROM de 1 BYTE cada uno lo que quiere decir que el número más alto que podemos guardar es el 255.

READ 5, pepe ; quiere decir leer la dirección 5 de la memoria EEPROM y guardar en la variable pepe, por consiguiente pepe se carga con el número 3.

WRITE 8, 137 ; quiere decir guardar en la dirección 8 de la memoria EEPROM el número 137, el dato anterior en este caso el 12 automáticamente se borra y se reescribe el número 137.

WRITE 7, pepe ; en este caso la variable pepe estaba cargada con el número 3, por consiguiente la dirección 7 de la memoria EEPROM se borra y se carga con el número 3.

NOTA: Estas declaraciones ya incluyen las PAUSAS necesarias para realizar la grabación, por lo que no es necesario colocar PAUSE 10 después de cada declaración, debe considerar también que la memoria EEPROM soporta 1'000.000 de ciclos de borrado/escritura.

cmcon=7	;cambiar a modo digital todo el puerto A
NUMERO VAR BYTE	;variable número para almacenar la tecla pulsada
R VAR BYTE	;variable r para hacer repeticiones
BIP VAR PORTA.0	;el portA.1 Bip (conectar chicharra activa)
LED VAR PORTA.1	;el portA.2 se llamará led
DOOR VAR PORTA.2	;el portA.3 conectar relé para la cerradura
A VAR PORTB.0	;nombres para los pines de las filas
B VAR PORTB.1	
C VAR PORTB.2	
D VAR PORTB.3	
UNO VAR PORTB.4	;nombres para los pines de las columnas
DOS VAR PORTB.5	
TRES VAR PORTB.6	
CUATRO VAR PORTB.7	
SETPRIME VAR BYTE	;variable para almacenar la 1era clave
SETSEGUN VAR BYTE	;variable para almacenar la 2da clave
SETERCER VAR BYTE	;variable para almacenar la 3era clave
SETCUART VAR BYTE	;variable para almacenar la 4ta clave
INICIANDO:	;programa del led para saber si está funcionando continúa ...
FOR R = 1 TO 2	

```

HIGH LED : HIGH BIP
PAUSE 1000
LOW LED : LOW BIP
PAUSE 150
NEXT

***** GUARDA LA CLAVE DE FABRICA *****
EEPROM 0, [ 1,2,3,4 ] :cargar la memoria EEPROM desde la dirección 0 en adelante
*****

RESET:
FOR R = 1 TO 3
HIGH LED : HIGH BIP
PAUSE 50
LOW LED : LOW BIP
PAUSE 50
IF (CUATRO=0)AND(UNO=0)THEN RESET :corresponden a teclas 7 y C
NEXT

READ 0,SETPRIME :leer el dato de la EEPROM 0 y guardar en setprime
READ 1,SETSEGUN :leer el dato de la EEPROM 1 y guardar en setsegun
READ 2,SETERCER :leer el dato de la EEPROM 2 y guardar en setercer
READ 3,SETCUART :leer el dato de la EEPROM 3 y guardar en setcuart

GOTO TECLAUNO :ir a comparar claves

GRABAUNO: :programa para cambiar la clave
GOSUB PTECLA : HIGH LED :espera a que suelte las teclas
GOSUB BARRIDO : GOSUB PTECLA :ir a barrido y retorna a un antirrebote
HIGH LED :mantener encendido el LED
WRITE 0,NUMERO :guardar en la EEPROM 0 el valor de número

GRABADOS:
GOSUB BARRIDO : GOSUB PTECLA :ir a barrido y retorna a un antirrebote
HIGH LED :mantener encendido el LED
WRITE 1,NUMERO :guardar en la EEPROM 1 el valor de número

GRABATRES:
GOSUB BARRIDO : GOSUB PTECLA :ir a barrido y retorna a un antirrebote
HIGH LED :mantener encendido el LED
WRITE 2,NUMERO :guardar en la EEPROM 2 el valor de número

GRABACUATRO:
GOSUB BARRIDO : GOSUB PTECLA :ir a barrido y retorna a un antirrebote
HIGH LED :mantener encendido el LED
WRITE 3,NUMERO :guardar en la EEPROM 3 el valor de número
GOTO RESET :ir a reset para cargar el nuevo valor en las variables

BARRIDO:
LOW A :sensar la fila A
IF UNO = 0 THEN NUMERO =1 :RETURN :tecla pulsada retorne cargada con 1
IF DOS = 0 THEN NUMERO =2 :RETURN :tecla pulsada retorne cargada con 2
IF TRES = 0 THEN NUMERO =3 :RETURN :tecla pulsada retorne cargada con 3
IF CUATRO=0 THEN NUMERO =10:RETURN :tecla pulsada retorne cargada con 10
HIGH A :continua ...

```



```

LOW B                                     ;sensar la fila B
IF UNO = 0 THEN NUMERO =4 : RETURN
IF DOS = 0 THEN NUMERO =5 : RETURN
IF TRES = 0 THEN NUMERO =6 : RETURN
IF CUATRO=0 THEN NUMERO =11: RETURN
HIGH B
LOW C                                     ;sensar la fila C
IF UNO = 0 THEN NUMERO =7 : RETURN
IF DOS = 0 THEN NUMERO =8 : RETURN
IF TRES = 0 THEN NUMERO =9 : RETURN
IF CUATRO=0 THEN NUMERO =12: RETURN
HIGH C
LOW D                                     ;sensar la fila D
IF UNO = 0 THEN NUMERO =14: RETURN
IF DOS = 0 THEN NUMERO =0 : RETURN
IF TRES = 0 THEN NUMERO =15: RETURN
IF CUATRO= 0 THEN NUMERO =13: RETURN
HIGH D
PAUSE 10
GOTO BARRIDO
; ***** programa de antirrebote de teclas *****
PTECLA:
HIGH LED : HIGH BIP                     ;genera sonido cada que se pulsa tecla
PAUSE 100                               ;duración 100 milisegundos
LOW LED : LOW BIP                       ;apagar sonido y led
ESPACIO:                                ;programa de antirrebote de teclas
IF UNO = 0 THEN ESPACIO                 ;si la tecla sigue pulsada ir espacio
IF DOS = 0 THEN ESPACIO                 ;si la tecla sigue pulsada ir espacio
IF TRES = 0 THEN ESPACIO                 ;si la tecla sigue pulsada ir espacio
IF CUATRO= 0 THEN ESPACIO               ;si la tecla sigue pulsada ir espacio
PAUSE 25
RETURN                                   ;retorna si se suelta las teclas

; ***** comparación de claves *****
TECLAUNO:
GOSUB BARRIDO                           ;ir a barrido y retornar con un valor
GOSUB PTECLA                             ;envía a un programa antirrebote para soltar tecla
IF numero = setprime THEN TECLADOS      ;si el número es igual a setprime
GOTO FALSO                               ;caso contrario ir a lazo falso
TECLADOS:
GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor
IF numero = setsegun THEN TECLATRES     ;si el número es igual a setsegun
GOTO FALSO1                              ;caso contrario ir a lazo falso
TECLATRES:
GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor
IF numero = setercer THEN TECLACUATRO   ;si el número es igual a setercer
GOTO FALSO2                              ;caso contrario ir a lazo falso
TECLACUATRO:
GOSUB BARRIDO :GOSUB PTECLA              ;ir a barrido y retornar con un valor

```

continúa ...

```

IF numero = setcuart THEN OPENGE ;si número es igual a setcuart conectar relé
GOTO FALSO3 ;caso contrario ir a lazo falso

OPENGE:
FOR R = 1 TO 2 ;2 pitos indica clave correcta
PAUSE 100
HIGH LED : HIGH BIP
PAUSE 100
LOW LED : LOW BIP
NEXT

HIGH DOOR ;se conecta el relé (abrir puerta)
PAUSE 1000 ;esperar 1 segundo
LOW DOOR ;desconectar relé
HIGH A : HIGH B : HIGH C : LOW D ;sensar solo la fila D
IF CUATRO = 0 THEN GRABAUNO ;corresponde a la tecla D para ir a GRABAR
GOTO TECLAUNO ;ir nuevamente a comparar las claves

; ***** lazos falsos teclas erróneas *****
FALSO:
GOSUB BARRIDO : GOSUB PTECLA ;estas teclas no comparan ninguna
FALSO1: ;clave solo espera que termine de
GOSUB BARRIDO : GOSUB PTECLA ;pulsar las 4 teclas y no hace nada
FALSO2:
GOSUB BARRIDO : GOSUB PTECLA

FALSO3:
FOR R = 1 TO 30 ;30 pitos indica clave incorrecta
PAUSE 150
HIGH LED : HIGH BIP
PAUSE 150
LOW LED : LOW BIP
HIGH A : HIGH B : HIGH D : LOW C ;sensar solo la fila C
IF (CUATRO=0)AND(UNO=0)THEN RESET ;corresponden a teclas 7 y C para resetear
NEXT

PANICO:
HIGH LED
PAUSE 500
LOW LED
PAUSE 500
HIGH A : HIGH B : HIGH D : LOW C ;sensar solo la fila C
IF (CUATRO=0)AND(UNO=0)THEN RESET ;corresponden a teclas 7 y C para resetear
GOTO PANICO ; queda en este lazo para siempre

END

```

Figura 5.7.4.1. cerradura EEPROM.bas Programa para hacer una cerradura electrónica codificada (1,2,3,4) en la que la clave se puede cambiar en la memoria EEPROM.

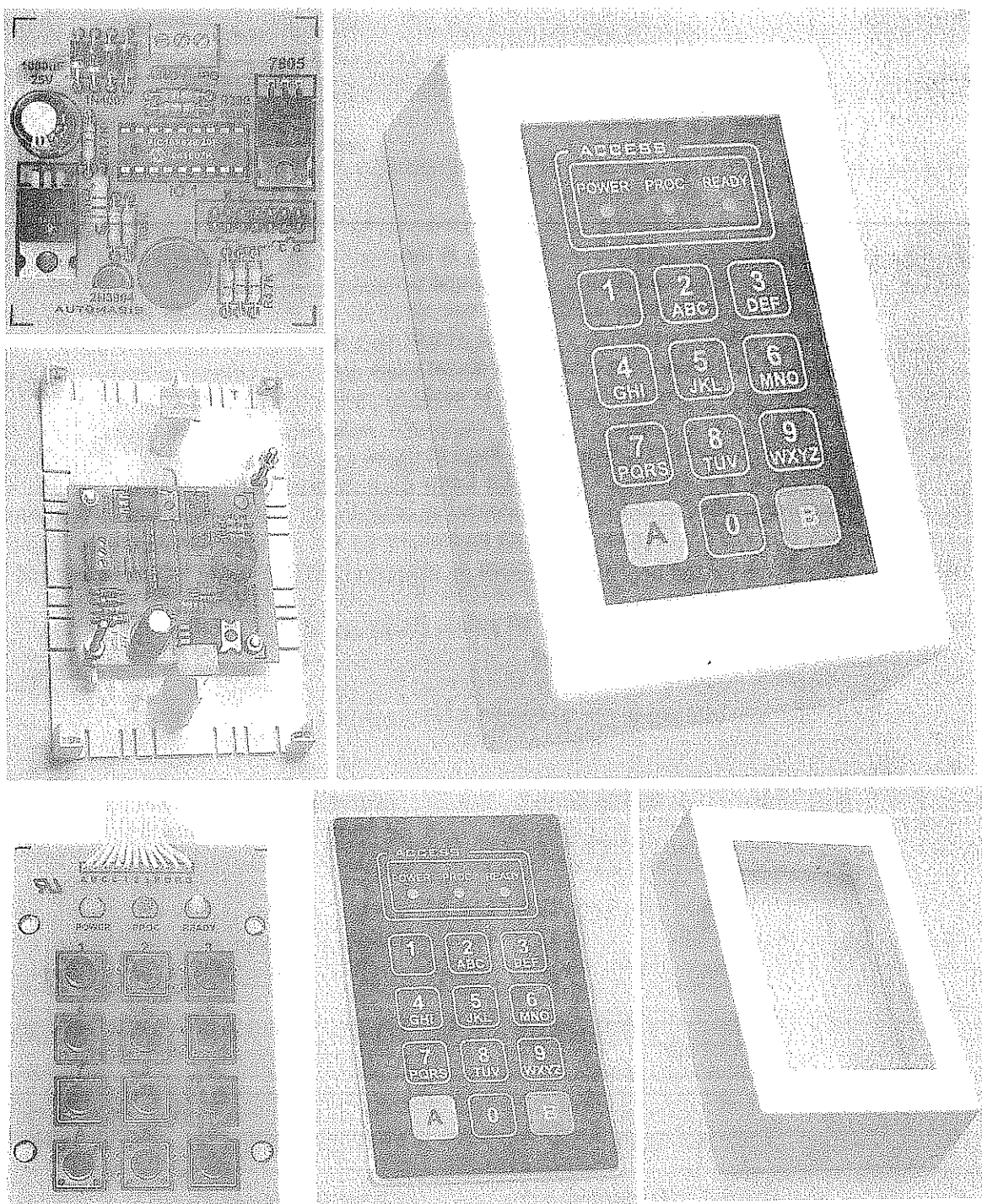


Figura 5.7.4.2. Fotografía de las partes que componen un control de accesos utilizado para abrir una cerradura eléctrica de 12 v. a través de un TIP110.

5.8 PROYECTOS CON MOTORES

5.8.1. MANEJO DEL PWM COMO VARIADOR DE VELOCIDAD DE UN MOTOR DC.

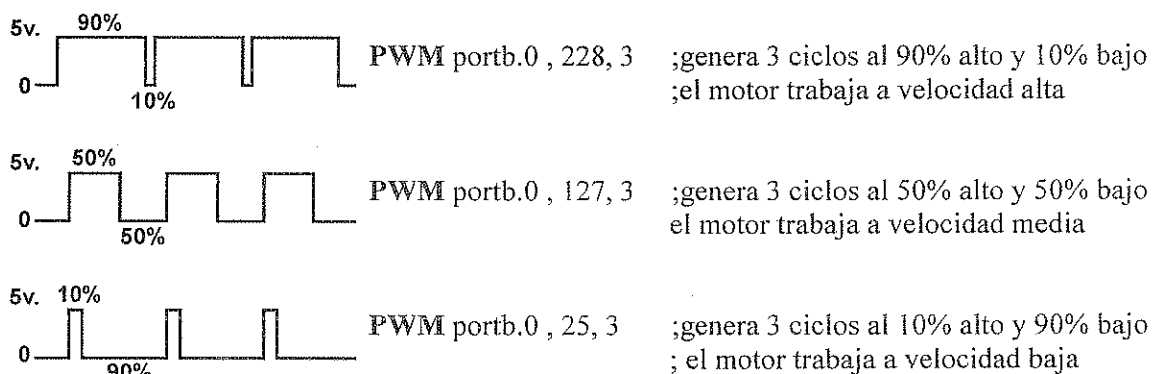
El PWM (Pulse Width Modulation) o modulación en ancho del pulso, tiene muchas aplicaciones, por ejemplo para atenuar la iluminación de un led, la iluminación del BACKLIGHT de un LCD, para variar la velocidad de un motor DC, que es lo que veremos en este caso.

El presente proyecto es un variador de velocidad de un motor DC de juguete que se alimenta a 5 voltios, su funcionamiento es de la siguiente manera:

Al momento de alimentar el circuito, el motor parte desde una velocidad media, es decir (FREC=125), al pulsar el botón (S) incrementa la variable en múltiplos de 25 y la velocidad del motor sube hasta llegar a (FREC=250), si seguimos pulsando la misma tecla, el LED permanecerá encendido, esto nos indica que ya llegó al límite, entonces pulsamos el botón (B), el cual hace que disminuya la velocidad del motor hasta llegar a (FREC=25), de igual manera si seguimos pulsando el botón (B) el LED permanecerá encendido. Si usted no dispone de un motor puede conectar a un LED directamente con una resistencia de $330\ \Omega$, igualmente podrá observar como baja o sube la intensidad del LED, la forma de utilizar el PWM es de la siguiente manera:

PWM portB.0, 127, 60 ; quiere decir sacar 60 pulsos PWM por el puerto B.0 al 50% en alto
; aproximadamente

La forma de la señal que sale por el PIC es similar a los siguientes gráficos:



Por consiguiente 0 representa 0% de ciclo útil y 255 el 100% de nivel alto, el largo de cada ciclo para un oscilador de 4MHZ es de **5 milisegundos** y para un oscilador de 20MHZ es de 1 milisegundo

MATERIALES.

- 3 resistencia de $4,7\ K\Omega$
- 1 resistencia de $330\ \Omega$
- 1 transistor TIP110
- 2 pulsadores normalmente abiertos
- 1 capacitor cerámico de $0,1\ \mu F$ (104)
- 1 LED rojo de 5 mm.
- 1 diodo rectificador 1N4007
- 1 motor de juguete.

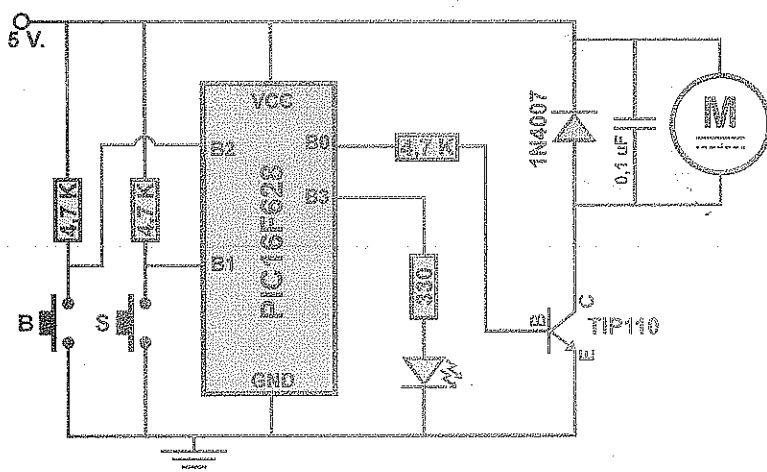


Figura 5.8.1.1. Conexión de un motor DC para manejar desde el PIC

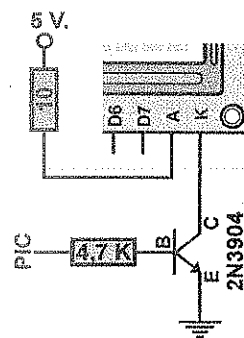


Figura 5.8.1.2. Conexión del Backlight del LCD, para manipular desde el PIC.

NOTA: El TIP110, puede manejar un motor de hasta 100 voltios DC a 8 Amperios, para el caso de querer utilizar un motor grande de AC, se recomienda utilizar un optoacoplador a la salida del puerto del PIC.

```

botsubir VAR portb.1           ;nombre para el pin B1
botbajar VAR portb.2           ;nombre para el pin B2
LED   VAR portb.3              ;nombre para el pin B3
FREC  VAR BYTE                 ;variable FREC tamaño 255
bandera VAR BIT                ;variable bandera de 1 bit
FREC = 125                     ;valor inicial para variable FREC

HIGH LED                       ;encender el led para saber que ya arrancó
PAUSE 500                      ;por medio segundo
LOW LED                        ;apagar el led

PROG:
  PWM PORTB.0,FREC,30           ;sacar PWM 30 ciclos de 125 señal útil
  LOW LED                      ;apagar el led
  IF botsubir =0 THEN SUBIR     ;si se pulsa el botón S ir a subir
  IF botbajar =0 THEN BAJAR     ;si se pulsa el botón B ir a bajar
  bandera =0                   ;bandera cargado con cero
  GOTO PROG

SUBIR:
  IF FREC > 249 THEN aviso      ;si supera a 249 ir a aviso
  IF bandera = 1 THEN prog      ;si la bandera esta en 1 salir
  HIGH LED                     ;encender el led
  bandera=1                    ;cargar la bandera con uno
  FREC = FREC +25               ;sumar 25 a la variable FREC
  GOTO PROG                     ;ir a prog

```

continúa ...

BAJAR: IF FREQ < 26 THEN aviso IF bandera = 1 THEN prog HIGH LED bandera=1 FREQ = FREQ -25 GOTO PROG	;si baja de 26 ir a aviso ;si la bandera esta en 1 salir ;encender el led ;cargar la bandera con uno ;restar 25 a la variable FREQ ;ir a prog
aviso: HIGH LED GOTO PROG END	;encender el led ;ir a prog

Figura 5.8.1.3. PWM motor.bas Programa para controlar la velocidad de un motor DC.

5.8.2. UN CONVERTOR D/A CON EL CI LM358.

Se puede hacer un pequeño convertidor de digital a analógico para el PWM con una resistencia y un capacitor, pero vamos a proponer realizarlo con el LM358 por sus mejores prestaciones, ya que lograremos mayor rango de voltaje (hasta 32 V.), pero en nuestro caso por motivos experimentales solo lo conectaremos a los 5 voltios de la misma fuente que está alimentado el PIC, en el caso de un PWM de 255, el LM358 tendrá en su salida 5 V., si sacamos un PWM de 127, tendremos 2,5 V., en definitiva los pulsos que ingresan al LM358 se convierten en salida analógica, desde 0 hasta 5 V.

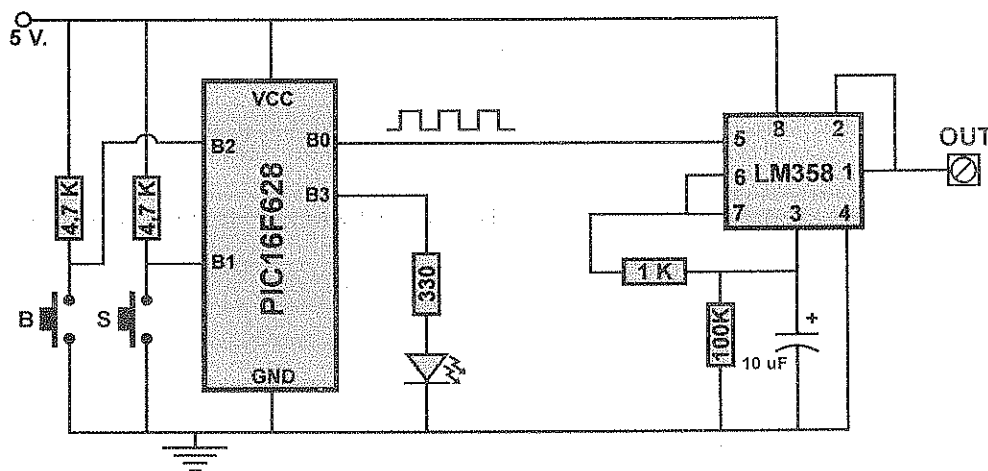


Figura 5.8.2.1. Conexión de un LM358 como convertor D/A para convertir el PWM en señal analógica de 0 a 5 V.

Se necesita un voltímetro para medir los niveles de voltaje en la salida, en cuanto al programa podemos utilizar el mismo del ejercicio anterior el PWM_motor.bas. A la salida del LM 358

podemos colocar un LED con su resistencia de 330 Ω . para poder observar su atenuación, también podemos colocar el circuito del motor de DC, con su capacitor y diodo de protección.

MATERIALES.

- 2 resistencia de 4,7 K Ω
- 1 resistencia de 330 Ω
- 1 resistencia de 1 K Ω
- 1 resistencia de 100 K Ω
- 1 capacitor electrolítico de 10 μ F/100V.
- 2 pulsadores normalmente abiertos
- 1 LED rojo de 5 mm.
- 1 CI. LM358.

5.8.3. LOS MOTORES PASO A PASO BIPOLARES Y UNIPOLARES.

Los motores paso a paso (PAP) son ideales para la construcción de mecanismos en donde se requieren movimientos muy precisos, como en robótica, en la tecnología aeroespacial, en maquinarias (tornos, fresadoras, bordadoras), en computadores (CD ROM, Disco duro, DVD, impresoras), etc. A diferencia de los motores de C.C. y los de C.A., los motores PAP tienen la ventaja de poder ser más precisos en cuanto a su velocidad, movimiento, enclavamiento y giros, y la señal que requieren para su funcionamiento es de naturaleza digital.

La manera de identificarlos a diferencia de los motores PAP bipolares de 4 hilos y 2 bobinas (ver figura 5.8.3.2.), es que los motores PAP unipolares tienen desde 5 hasta 8 alambres (ver figura 5.8.3.3.) y su funcionamiento es mucho más simple que los motores PAP bipolares, los cuales necesitan un integrado L293 que dispone de 2 puentes H (H-Bridge) o por lo menos debemos hacer un arreglo de 8 transistores, (4 PNP y 4 NPN).

En cuanto al voltaje de alimentación existen desde 1,3V., 1,9V., 4,5V., 5V., 12V. y 24 V., la corriente de consumo de un motor puede estar desde 300mA hasta 3 A.

De acuerdo a la aplicación que deben realizar los motores PAP tienen diferentes grados de precisión como muestra la siguiente tabla:

Grados que gira por impulso θ	Nro. de pasos para llegar a 360°
0,72°	500
1,8°	200
3,75°	96
7,5°	48
15°	24
90°	4

Figura 5.8.3.1. Tabla del número de pasos que debe dar un motor PAP para llegar a dar una vuelta completa, según su ángulo de giro.

El circuito de control para los motores PAP unipolares sea de 5, 6, 8 hilos, es muy sencillo, podemos utilizar un buffer ULN2803 o 4 transistores TIP110 con 4 diodos de protección, para

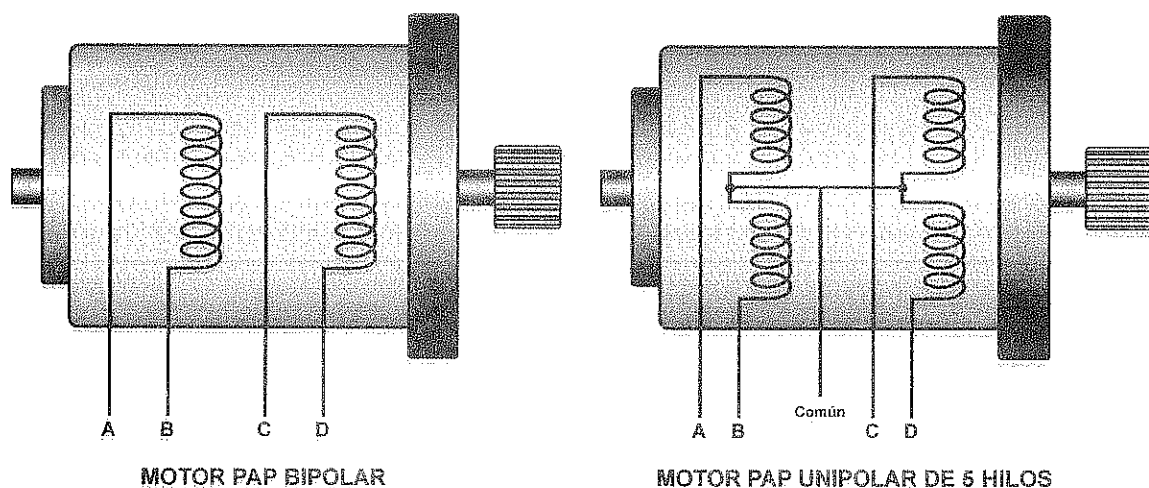


Figura 5.8.3.2. Diferencia entre un motor PAP bipolar y un motor PAP unipolar de 4 bobinas.

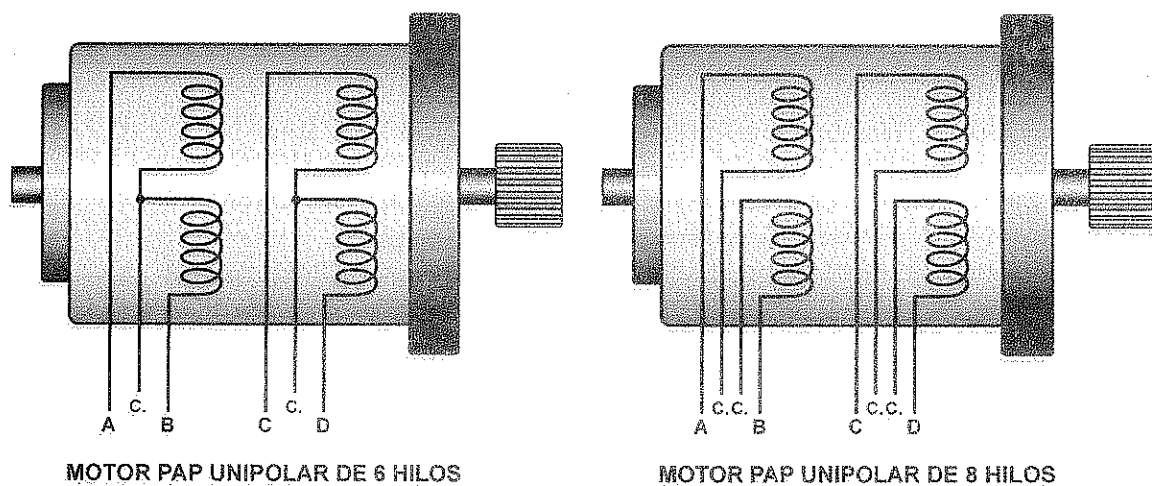


Figura 5.8.3.3. Motores PAP unipolar de 4 bobinas de 6 y 8 hilos, los cuales se los puede configurar como bipolares.

que empiece a girar basta con dar una secuencia de pulsos con una duración de 5 milisegundos a cada bobina como muestra la figura 5.8.3.4, mientras que para un motor bipolar se debe invertir la polaridad de cada bobina para que este pueda generar un paso como lo muestra la figura 5.8.3.6.

BOBINA	P1	P2	P3	P4
A	+V	gnd	gnd	gnd
C	gnd	+V	gnd	gnd
B	gnd	gnd	+V	gnd
D	gnd	gnd	gnd	+V

Figura 5.8.3.4. Tabla de la secuencia de energizado de bobinas para un motor PAP unipolar para que este gire en sentido antihorario.

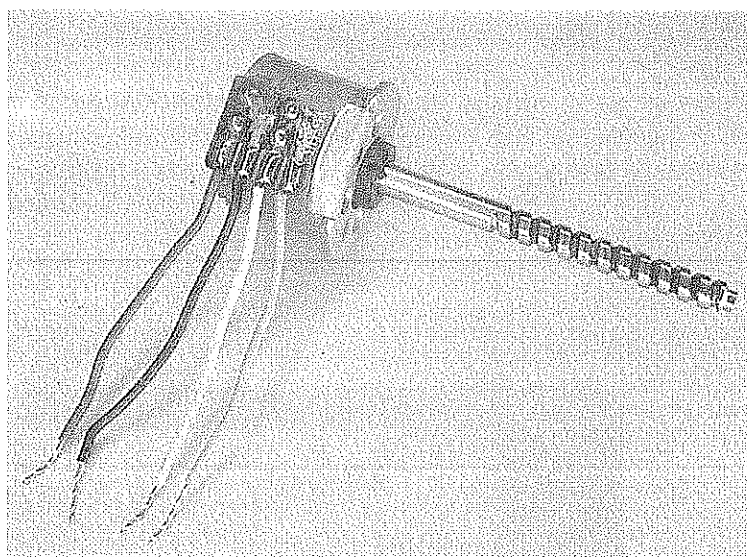


Figura 5.8.3.5. Fotografía de un motor PAP bipolar de la diske-tera de un computador, observen que tiene solo 4 hilos.

PASO	A	B	C	D
1	V+	V-	V+	V-
2	V+	V-	V-	V+
3	V-	V+	V-	V+
4	V-	V+	V+	V-

Figura 5.8.3.6. Tabla de la secuencia de conmutación para un motor PAP bipolar.

Uno de los mayores inconvenientes a la hora de trabajar con los motores PAP en especial los unipolares es la de poder identificar cual es la bobina A, B, C, y D, para esto simplemente medimos las resistencias de cada una de las bobinas, por ejemplo tomemos el caso de un motor unipolar de 6 hilos cuyos datos de la placa dice:

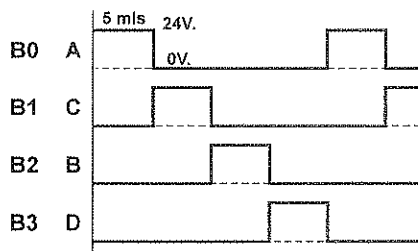
STEPPING MOTOR CBK5-12	
VOLT	24 V
COIL	22 Ω
DEG/STEP	7.5 \circ

Como podemos ver es un motor paso a paso de 7,5 grados de movimiento con una alimentación requerida de 24 V. y cuyas bobinas tienen una resistencia de 22 Ω cada una.

Para identificar qué bobina es la A, debemos buscar el alambre de color amarillo, la B el de color naranja, C el de color negro, el D el de color café, y los dos rojos son los comunes. Para el caso de que no coincidan con esta gama de colores, debemos medir la resistencia entre un cable y otro cable, los que marquen 22 Ω son bobinas común y un terminal y si marca 44 Ω son los terminales A y B o C y D, una vez ubicado cuales son los comunes, estos podemos unirlos en un solo cable y colocando el motor de la forma que ilustra el gráfico 5.8.3.3, ya podemos asignar los lugares de cada bobina, otra manera de ubicarlos es haciendo pruebas, si los cables o solo un par de cables están conectados incorrectamente, el motor no gira y en su lugar permanece temblando, en este caso pruebe cambiando los cables hasta que el motor empiece a girar.

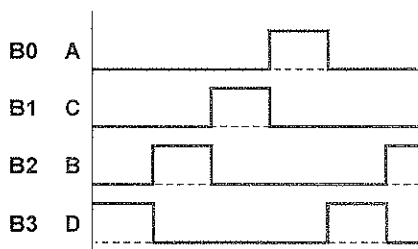
5.8.4. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA WAVE DRIVE.

Como práctica básica para introducirnos en el manejo de motores PAP vamos a hacer un programa que genere una revolución completa a un motor de 7.5 grados a 24 voltios en secuencia wave drive o secuencia por ola, esta es la forma más fácil de manejar un motor, consiste en energizar una sola bobina a la vez, A, C, B, y por último la D, a continuación veremos la tabla de energizado para conseguir que el motor gire en ambos sentidos.



Bobina	P 1	P 2	P 3	P 4
A	1	0	0	0
C	0	1	0	0
B	0	0	1	0
D	0	0	0	1

Figura 5.8.4.1. Tabla de energizado de bobinas en secuencia por ola para giro antihorario.



Bobina	P 1	P 2	P 3	P 4
A	0	0	0	1
C	0	0	1	0
B	0	1	0	0
D	1	0	0	0

Figura 5.8.4.2. Tabla de energizado de bobinas en secuencia por ola para giro horario.

Como podemos ver solo necesitamos activar un pin a la vez durante 5 milisegundos, si ponemos 10 milisegundos esto hará que el motor gire más despacio, pero menos de 4 milisegundos, no serán suficiente para generar el paso y el motor se quedará temblando, adicionalmente se debe poner diodos de protección del colector de cada transistor al voltaje positivo que esté conectado el cable común del motor, esto para proteger al PIC del efecto inductivo que genera el motor. El programa que haremos a continuación hace girar 360° en sentido antihorario, se detendrá por un segundo y luego girará otros 360° en sentido horario, y así indefinidamente, como este es un motor de 7,5° de giro, necesitaremos repetir la secuencia de los pasos 12 veces, luego pruebe con 6 veces y verá que el motor gira 180°.

MATERIALES.

- 4 resistencia de 4,7 K Ω
- 4 diodos rectificadores 1N4007
- 4 transistores TIP110
- 1 motor PAP unipolar de cualquier voltaje desde 5 hilos a 8 hilos.

Los motores PAP unipolares de 7,5° los podemos conseguir comúnmente en algunas impresoras de las que ya no utilizamos, estas se encuentran en el mecanismo que mueven los rodillos del alimentador de papel, y algunas impresoras tienen internamente hasta 2 motores PAP.

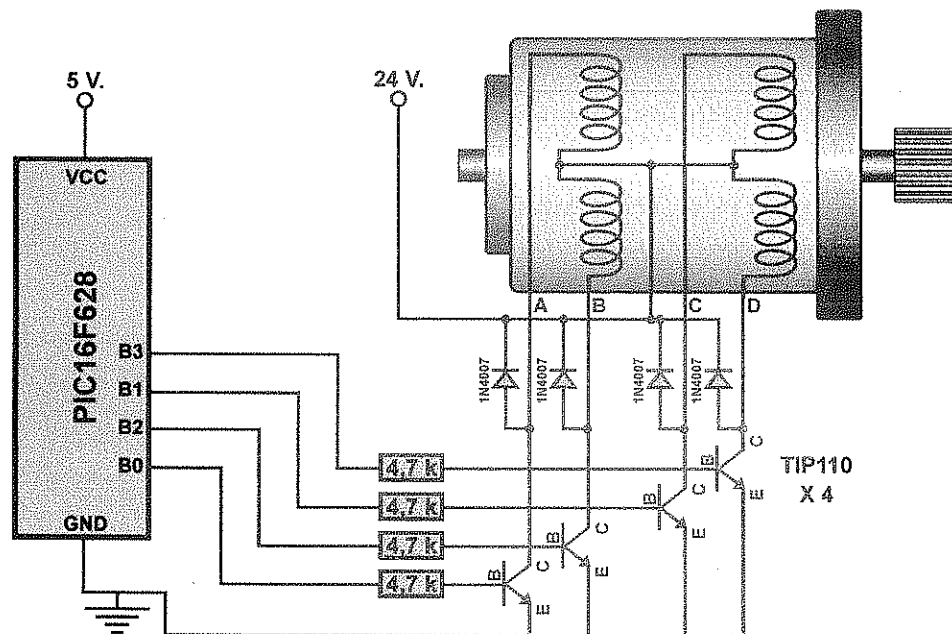


Figura 5.8.4.3. Conexión de un motor PAP unipolar a las salidas del PIC.

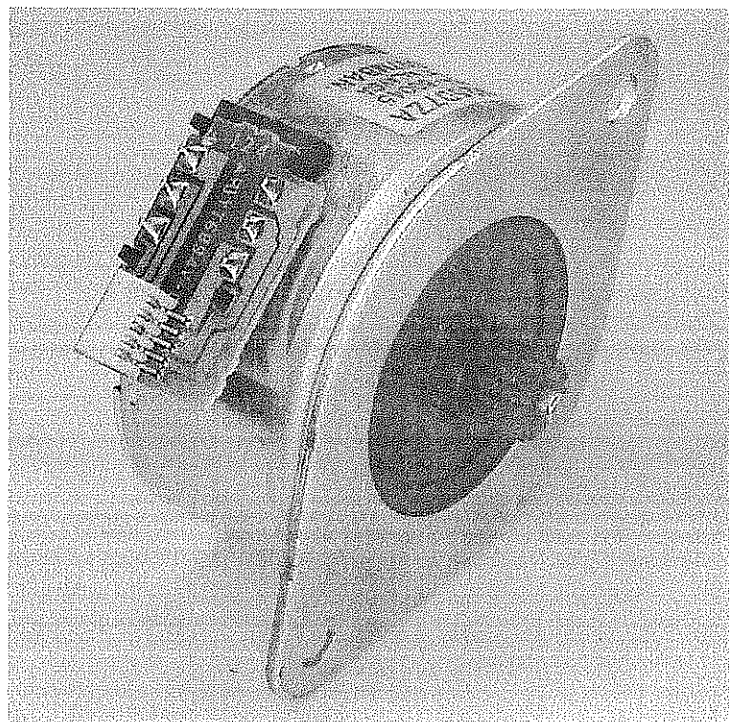


Figura 5.8.4.4. Fotografía de un motor PAP de una impresora de inyección a tinta.

trish=0	;hacer salidas el puerto b
x VAR BYTE	;variable x de 255
antihorario:	
FOR x = 1 TO 12	;12 veces repetir secuencia de giro antihorario
portb=%0001	;energiza bobina A
GOSUB timer	;espera 5 mls
portb=%0010	;energiza bobina C
GOSUB timer	;espera 5 mls
portb=%0100	;energiza bobina B
GOSUB timer	;espera 5 mls
portb=%1000	;energiza bobina D
GOSUB timer	
NEXT	
PAUSE 1000	;espera 1 segundo
FOR x = 1 TO 12	;12 secuencias para giro en sentido horario
portb=%1000	;energiza bobina D
GOSUB timer	;espera 5 mls
portb=%0100	;energiza bobina B
GOSUB timer	;espera 5 mls
portb=%0010	;energiza bobina C
GOSUB timer	;espera 5 mls
portb=%0001	;energiza bobina A
GOSUB timer	;espera 5 mls
NEXT	
PAUSE 1000	;espera 1 segundo
GOTO antihorario	
timer:	
PAUSE 5	;pausa de 5 milisegundos
RETURN	
END	

Figura 5.8.4.5. Motor PAP sec-OLA.bas Programa para hacer girar 360° en ambos sentidos a un motor PAP unipolar .

5.8.5. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA FULL STEP.

También conocida como secuencia por paso completo, este es la manera que recomiendan los fabricantes, debido a que siempre se encuentran energizadas 2 bobinas, se obtiene un alto torque de paso y de retención, y consume el 40% más de corriente que el caso anterior.

El siguiente ejercicio hace girar el motor continuamente, noten además la fuerza que tiene tratando de detener el giro con sus dedos.

En cuanto a los materiales y el diagrama de conexión utilizaremos los mismos del ejercicio anterior.

A continuación mostraremos la grafica de secuencia de pasos y energizado para el manejo del motor a paso completo.

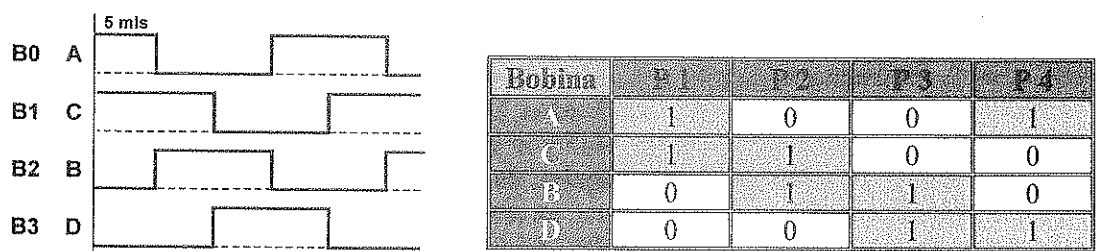


Figura 5.8.5.1. Tabla de energizado de bobinas para la secuencia de paso completo.

```

trsb=0                                ;hacer salidas el puerto b
antihorario:
    portb=%0011                        ;energiza bobina A y C
    PAUSE 5                            ;espera 5 mls
    portb=%0110                        ;energiza bobina C y B
    PAUSE 5                            ;espera 5 mls
    portb=%1100                        ;energiza bobina B y D
    PAUSE 5                            ;espera 5 mls
    portb=%1001                        ;energiza bobina D y A
    PAUSE 5                            ;espera 5 mls
    GOTO antihorario                  ;continuar girando
END

```

Figura 5.8.5.2. Motor PAP sec-paso compl.bas Programa para hacer girar continuamente en secuencia paso completo.

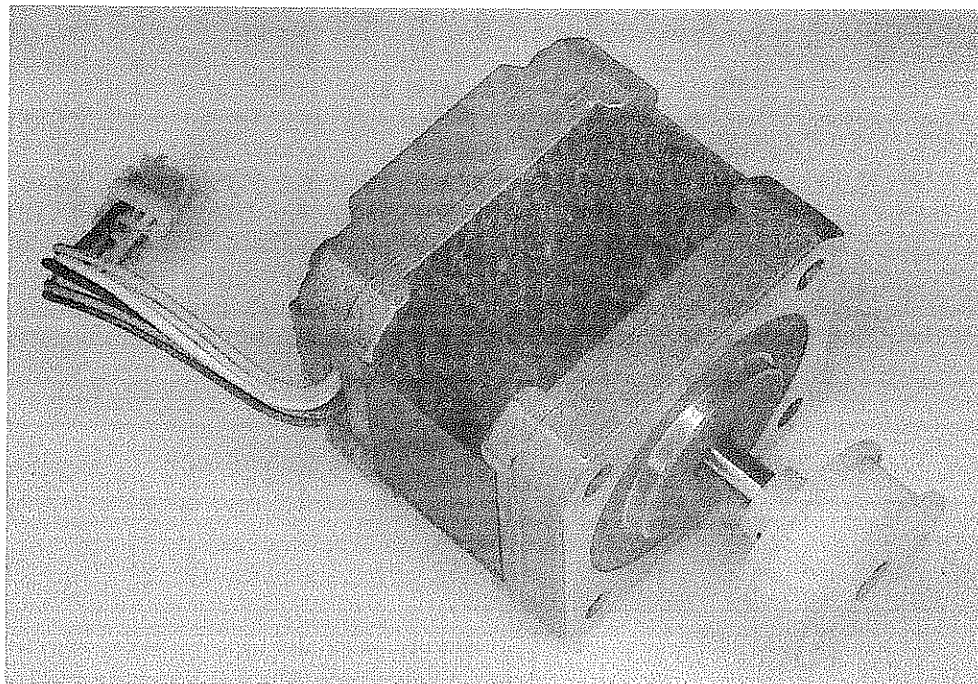


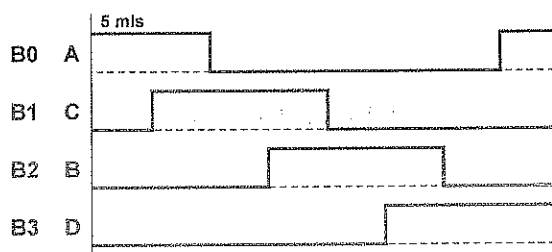
Figura 5.8.5.3. Fotografía de un motor PAP unipolar de 6 hilos a 6 V. que consume 760 mA.

5.8.6. MANEJO DE UN MOTOR PASO A PASO EN SECUENCIA HALF STEP.

También conocida como secuencia a medio paso, este es una combinación de las 2 secuencias anteriores, se energiza 2 bobinas, luego 1 bobina, luego otra vez 2 bobinas y así alternadamente, como resultado el rotor avanza medio paso por cada pulso de excitación, la ventaja de esta secuencia es la disminución del avance angular, de $7,5^\circ$ a $3,75^\circ$, por consiguiente para girar una vuelta completa se necesita el doble de pasos, en este caso 96, lo que veremos en el programa será 8 secuencias que multiplicado por 12, nos dará 96 y esto equivale a 360° .

En cuanto a los materiales y el diagrama de conexión utilizaremos los mismos del ejercicio anterior.

A continuación mostraremos la grafica de secuencia de pasos y energizado para el manejo del motor a medio paso.



Bobina	P1	P2	P3	P4	P5	P6	P7	P8
A	1	1	0	0	0	0	0	1
C	0	1	1	1	0	0	0	0
B	0	0	0	1	1	1	0	0
D	0	0	0	0	0	1	1	1

Figura 5.8.6.1. Tabla de energizado de bobinas para la secuencia de medio paso.

trisb=0	;hacer salidas el puerto b
REPT VAR BYTE	;crear variable REPT de 255
FOR REPT = 1 TO 12	;repetir 12 veces (360 grados)
portb=%0001	;energizar bobina A
PAUSE 5	;retardo de 5 mls
portb=%0011	;energizar bobina A y C
PAUSE 5	;retardo de 5 mls
portb=%0010	;energizar bobina C
PAUSE 5	;retardo de 5 mls
portb=%0110	;energizar bobina C y B
PAUSE 5	;retardo de 5 mls
portb=%0100	;energizar bobina B
PAUSE 5	;retardo de 5 mls
portb=%1100	;energizar bobina B y D
PAUSE 5	;retardo de 5 mls
	continúa ...

portb=%0100	;energizar bobina B
PAUSE 5	;retardo de 5 mls
portb=%1100	;energizar bobina B y D
PAUSE 5	;retardo de 5 mls
portb=%1000	;energizar bobina D
PAUSE 5	;retardo de 5 mls
portb=%1001	;energizar bobina A y D
PAUSE 5	;retardo de 5 mls
NEXT	;siguiente repetición
END	;fin del movimiento

Figura 5.8.6.2. Motor PAP sec-medio paso.bas Programa para hacer girar 360°, en sentido antihorario con un avance de medio paso.

NOTA: En este caso observarán que al terminar de dar la vuelta completa, quedan energizadas las bobinas A y D, por lo tanto el motor quedará retenido fuertemente y empezará a sobrecalentarse, para que esto no suceda adicionen después del NEXT la línea portb=%0000, con esto no queda ninguna bobina energizada y por consiguiente notarán que el rotor gira libremente.

5.9 COMUNICACIÓN

5.9.1. ¿QUÉ ES LA COMUNICACIÓN SERIAL?

Existen dos formas de realizar una comunicación binaria, la paralela y la serial. La comunicación paralela como por ejemplo la comunicación del PIC con el CI. 7447 del ejercicio 5.4.1., en donde los datos viajan simultáneamente a través de los 4 hilos, tiene la ventaja de que la transferencia de datos es más rápida, pero el inconveniente es que necesitamos un cable por cada bit de dato, lo que encarece y dificulta el diseño de las placas, otro inconveniente es la capacitancia que genera los conductores por lo que la transmisión se vuelve defectuosa a partir de unos pocos metros.

La comunicación serial en cambio es mucho más lenta debido a que transmite bit por bit pero tiene la ventaja de necesitar menor cantidad de hilos, y además se puede extender la comunicación a mayor distancia, por ejemplo; en la norma RS232 a 15 mts., en la norma RS422/485 a 1200mts y utilizando un MODEM, pues a cualquier parte del mundo.

Existen dos formas de realizar la comunicación serial: la sincrónica y la asincrónica, la diferencia entre estas dos formas de comunicación es que la comunicación sincrónica además de la línea para la transmisión de datos, necesita otra línea que contenga los pulsos de reloj, estos a su vez indican cuando un dato es válido. Por otra parte la comunicación serial asincrónica no necesita pulsos de reloj, en su lugar utiliza mecanismo como referencia tierra (RS232) o voltajes diferenciales (RS422/485), en donde la duración de cada bit es determinada por la velocidad de transmisión de datos que se debe definir previamente entre ambos equipos.

5.9.2. MODOS DE TRANSMISIÓN DE DATOS.

Se incluye este literal para poder entender mejor las prácticas que más adelante realizaremos, pues mencionaremos algunas palabras que podría encontrar su significado en este literal.

Los modos de transmisión de datos se dividen en cuatro tipos y estos son:

5.9.2.1. Simplex. Se dice a la transmisión que puede ocurrir en un solo sentido, sea sólo para recibir o sólo para transmitir. Una ubicación puede ser un transmisor o un receptor, pero no ambos a la vez, un ejemplo claro es la radiodifusión, en donde la estación es el transmisor y los radios son los receptores.

5.9.2.2. Half-duplex. Se refiere a la transmisión que puede ocurrir en ambos sentidos pero no al mismo tiempo, en donde una ubicación puede ser un transmisor y un receptor, pero no los dos al mismo tiempo, un ejemplo son los llamados radios WALKING TALKING, en donde un operador presiona el botón y habla, luego suelta el botón y el otro usuario presiona el botón para contestar.

5.9.2.3. Full-duplex. Se dice a la transmisión que puede ocurrir en ambos sentidos y al mismo tiempo, también se los conoce con el nombre de líneas simultaneas de doble sentido, una ubicación puede transmitir y recibir simultáneamente, siempre y cuando la estación a la que está transmitiendo también sea la estación de la cual está recibiendo un ejemplo es la telefonía móvil.

5.9.2.4. Full/full-duplex. Con este modo de transmisión es posible transmitir y recibir simultáneamente, pero no necesariamente entre las dos ubicaciones, es decir una estación puede transmitir a una segunda estación y recibir de una tercera estación al mismo tiempo. Esta transmisión se utilizan casi exclusivamente con circuitos de comunicación de datos.

5.9.3. COMUNICACIÓN SERIAL RS232.

La norma RS232 se incluye actualmente en los computadores, conocido como puerto serial y sirve para comunicarse con otras computadoras además del mouse, programadores, impresoras, etc. A continuación veremos un gráfico que muestra la forma de comunicación serial.

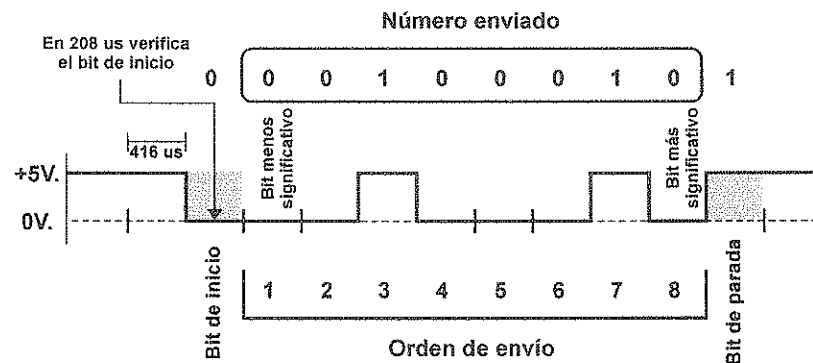


Figura 5.9.3.1. Estructura de un dato que se envía serialmente a 2400,8N1, (2400bits /seg, sin paridad, 8 bits de dato y 1 bit de parada), correspondiente al número 68 carácter ASCII de "D" (%01000100), el tiempo de un bit es de 416 μ s., por lo que el receptor revisa el bit de arranque después de 208 μ s., y luego cada 416 μ s.

Como podemos ver la señal permanece en un nivel lógico alto mientras no realiza ninguna transferencia de datos. Para empezar a transmitir datos el transmisor coloca la línea en nivel bajo durante el tiempo de un bit (416 μ s para 2400bits/s), este se llama el bit de arranque, a continuación empieza a transmitir con el mismo intervalo de tiempo los bits de datos, que pueden ser de 7 u 8 bits, comenzando por los bits menos significativos y terminando por los más significativos. Al final de la transmisión de datos se envía el bit de paridad, si estuviera activa esta opción y por último los bits de parada, que pueden ser 1 o 2, después de esto la línea vuelve a un estado lógico alto, y el transmisor está listo para enviar el siguiente dato.

Como el receptor no está sincronizado con el transmisor desconoce el momento en que empieza la transmisión, por lo que siempre debe estar en espera del cambio de estado o sea el bit de arranque, una vez que se da este bit, medio bit después vuelve a verificar si está en bajo, si no lo está no lo recibe ya que pudo ser ocasionado por un ruido en la línea, caso contrario si el estado sigue siendo bajo, empieza a recibir la transmisión hasta el bit de parada.

Para que la lectura de los datos sea correcta, ambos equipos deben estar configurados a la misma velocidad y demás parámetros y no exceder más allá de los 2 metros, pasado esta distancia los datos recibidos pueden no ser los correctos debido a la pérdida de voltaje en el cable, ruido, etc. Para distancias mayores existe el protocolo RS232, cuyos niveles de voltaje están establecidos de la siguiente manera: para señal 1 lógica (-5V a -15V) en el transmisor y (-3V a -25V) en el receptor, para señal 0 lógica (+5V a +15 V) en el transmisor y (+3V a +25V) en el receptor, es decir una lógica inversa.

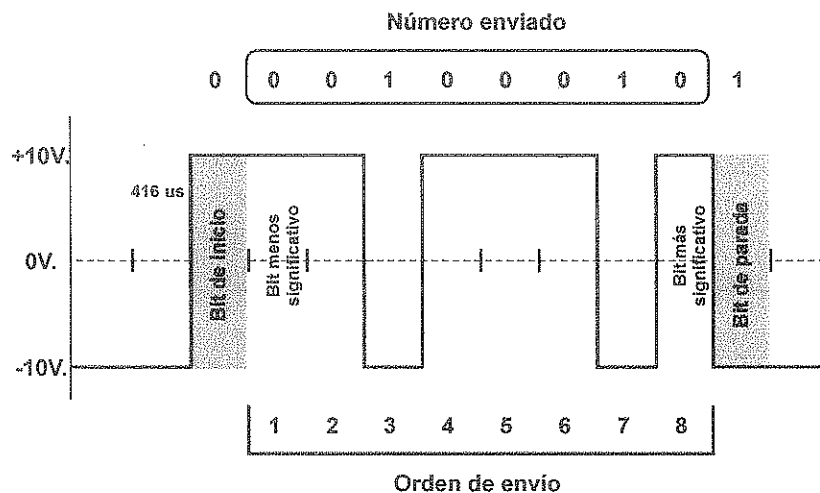


Figura 5.9.3.2. Comunicación serial con la norma RS232, el dato enviado es el mismo que el de la figura 5.9.3.1., con la diferencia que la lógica es inversa, 1 equivale a -10 y 0 a $+10$.

5.9.4. COMUNICACIÓN SERIAL PIC A PC.

Una vez comprendido la teoría de la comunicación serial y su protocolo RS232, haremos un ejercicio de comunicación serial asincrónica modo simplex, que consiste en enviar datos, más específicamente los caracteres ASCII de la palabra "DOG", a través de un cable y directamente desde el PIC al PC, a 2400 bits/seg., a 8 bits de datos, sin paridad, y 1 bit de parada. Como sabemos el computador tiene al menos un puerto serial, con la norma RS232, por lo tanto debemos simular esos voltajes desde el PIC, esto lo conseguimos enviando 0 para representar el 1 lógico y 5V. para representar el 0 lógico, para esto existe la declaración **SEROUT**.

LA DECLARACIÓN SEROUT. Esta declaración sirve para enviar datos seriales en un formato standard asincrónico usando 8 bits de dato, sin paridad y 1 stop bit, (8N1), y para poder utilizarlo

SEROUT puerto B.1, N2400,["D"] ;enviar el caracter ASCII "D"por el puerto B1 a 24008N1, en dato invertido (figura 5.9.4.1.).

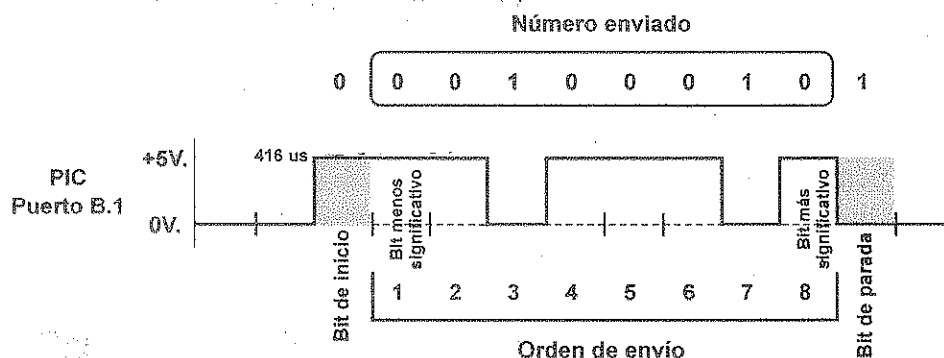


Figura 5.9.4.1. Esquema del dato enviado por el PIC simulando la norma RS232, noten que es muy similar al esquema 5.9.3.2. pero con diferentes niveles de voltaje.

debemos incluir al comienzo del programa la siguiente línea:

```
INCLUDE "modedefs.bas" ;incluir el programa modedefs.bas(modos de comunicación)
```

Esto significa incluir el programa modedefs.bas en esta línea, aquí se encuentran algunos de los parámetros para las comunicaciones, por ejemplo en nuestro caso las velocidades de transmisión que son: para dato invertido N300, N1200, N2400, N9600, y para dato verdadero: T300, T1200, T2400, T9600. Los datos invertidos por ejemplo el N2400, quiere decir que un 1 lógico vale 0V. y un 0 Lógico vale 5V. (ver figura 5.9.4.1.), en cambio para dato verdadero por ejemplo el T2400 el 1 lógico vale 5V. y el 0 lógico vale 0V. esta señal saldría muy similar al de la figura 5.9.3.1. y este es el que se utiliza para manejar con el CI. MAX232, el cual ya veremos más adelante.

Este comando **INCLUDE** podemos utilizarlo para nuestros propios programas por ejemplo si ponemos **INCLUDE** "freqout.bas", se incluirá el sonido para un parlante por el puerto B0 que durará 2 segundos.

MATERIALES.

- 1 conector DB9 hembra con su respectivo cajetín
- 1 resistencia de 220 Ω
- 1,5 mts de cable de 2 pares para transmisión de datos
- 1 led rojo 5mm
- 1 resistencia 330 Ω .

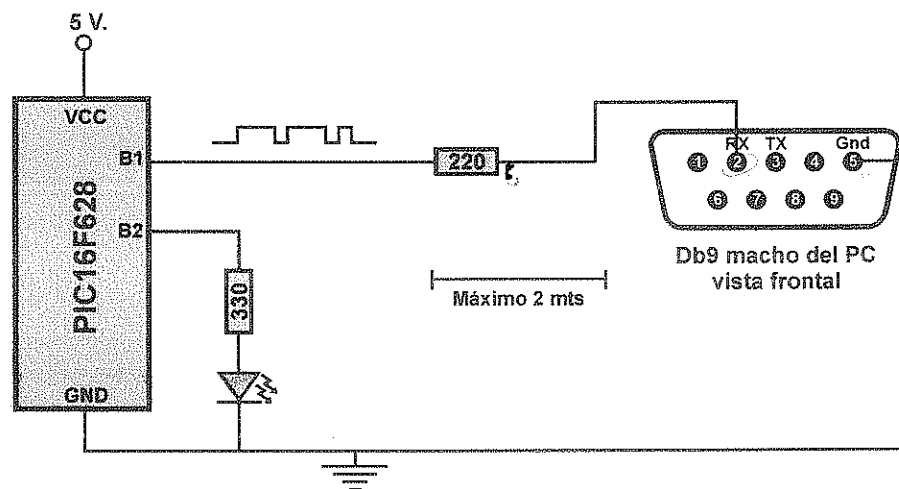


Figura 5.9.4.2. Diagrama de conexión del PIC para enviar datos al PC sin el CI. MAX232.

```
INCLUDE "modedefs.bas" ;incluyen los modos de comunicación
```

Inicio:

```
HIGH portb.2
```

```
PAUSE 1000
```

```
LOW portb.2
```

```
PAUSE 500
```

```
;led prueba de funcionamiento
```

continúa...


```

SEROUT portb.1, N2400,["DOG"] ; enviar serialmente a 2400N1 los caracteres "DOG"

GOTO inicio ; volver a repetir el proceso
END

```

Figura 5.9.4.3. *Serout PIC-PC.bas Programa para transmitir serialmente desde el PIC a un PC sin utilizar el CI. MAX232.*

Una vez que tenemos listo el proyecto necesitamos una ventana de comunicación serial como el Hyper terminal o la misma ventana de comunicación serial que dispone microcode, para esto presionamos en la pantalla de microcode F4 o el botón  y configuramos los parámetros que necesitamos.

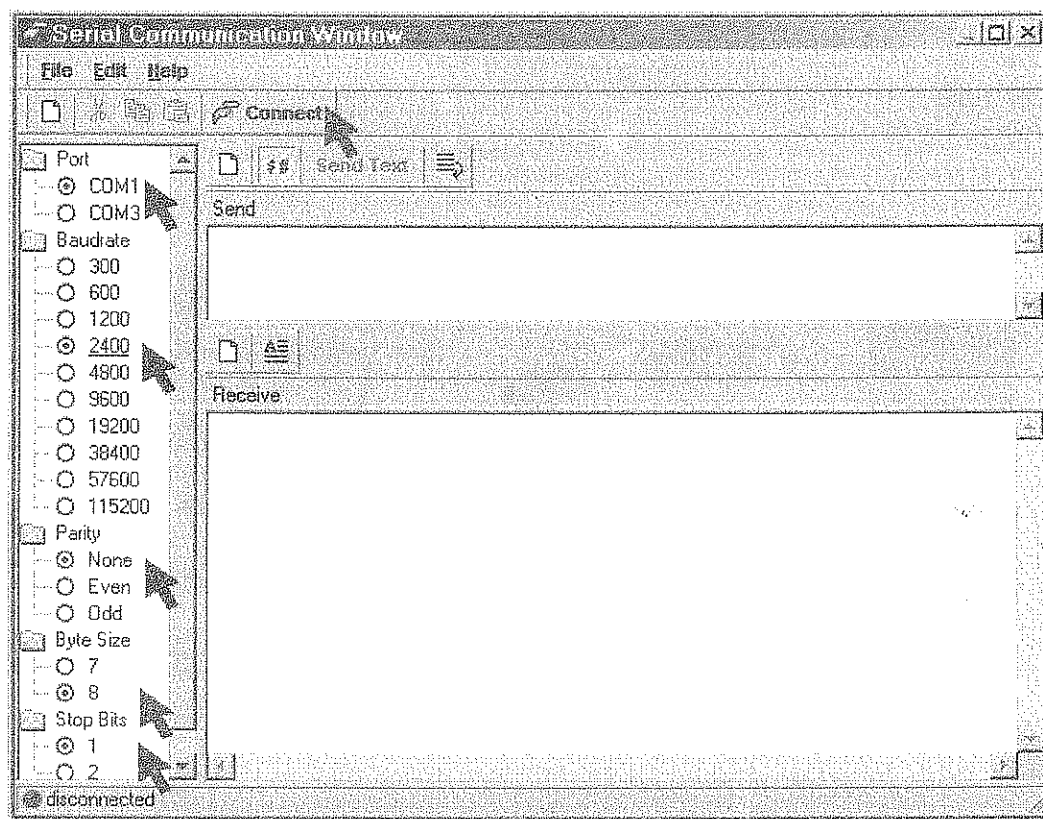


Figura 5.9.4.4. *Pantalla de la ventana de comunicación serial que dispone microcode.*

Para este ejercicio debemos primero seleccionar el puerto con que vamos a utilizar, luego la velocidad que se transmite el dato, en este caso a 2400 baud, luego paridad ninguna, 8 bits de datos y 1 stop bit, una vez que estemos listos para iniciar la comunicación presionamos el botón connect, notarán en la parte inferior izquierda que decía disconnected cambiará como lo demuestra la siguiente figura:

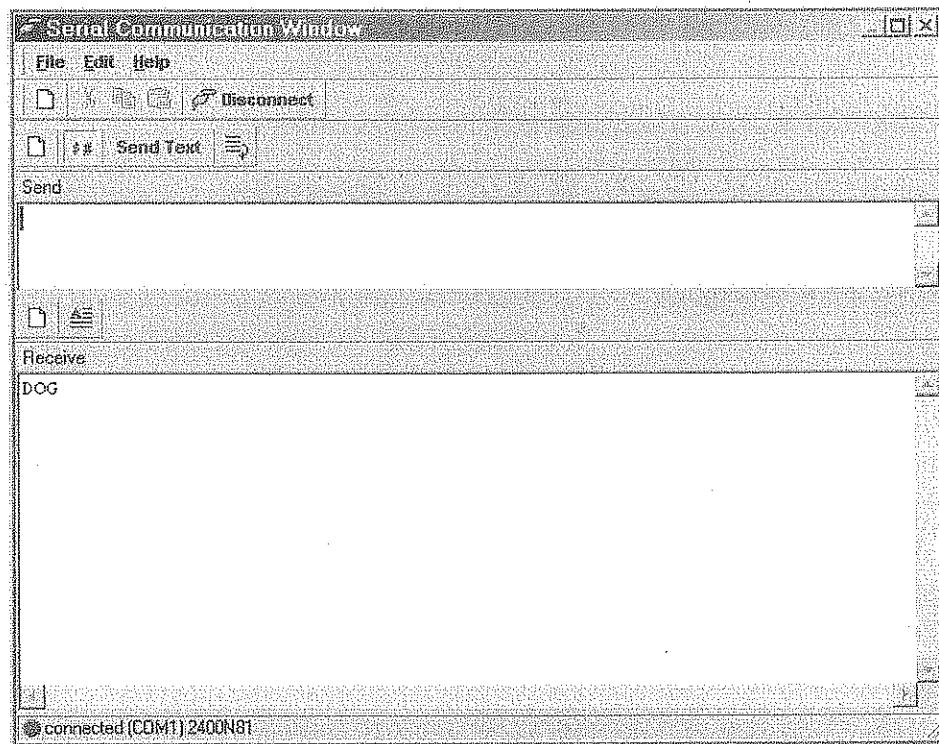


Figura 5.9.4.5. Pantalla de la ventana de comunicación serial activada.

Cuando la ventana está activa sale un mensaje en la parte inferior izquierda connected, el com que se está utilizando y los parámetros de comunicación (2400N81), encienda el micro PIC y después de apagarse el LED del puerto B.2 deberá salir el texto enviado en el cuadro que dice Receive, como el programa está en un lazo sin fin el texto DOG seguirá saliendo continuamente, si deseamos limpiar el texto presionamos la hoja en blanco que está justo encima de la palabra Receive.

NOTA: Es importante dejar cerrando esta ventana presionando donde dice Disconnect, ya que si se la deja abierto este com no se puede utilizar para grabar micros o cualquier otra aplicación.

IMPORTANTE. Si los mensajes que visualiza el computador son erróneos, pruebe enviando una sola letra o cambiando la transmisión a N300, si aún sigue saliendo datos erróneos utilice un oscilador externo desde 4MHZ en adelante, y si con estos cambios aún sigue fallando, pues obligadamente necesitaremos utilizar el CI. MAX232 que veremos más adelante.

NOTA: El nuevo PIC16F628A que está ingresando actualmente al mercado, obligadamente necesita oscilador externo de 4MHZ a 16 MHZ para poder realizar estas prácticas.

5.9.5. COMUNICACIÓN SERIAL PC A PIC.

Si usted tuvo problemas con el envío de datos del PIC al PC, este proyecto de seguro le funcionará, se trata de enviar datos desde el PC al PIC, por lo que es de suponerse los voltajes serán desde -10V. hasta +10V. y la distancia podemos extenderlo hasta 15 mts. sin ningún problema, como la conexión es directamente al PIC debemos colocar una resistencia de 22K para

LA DECLARACIÓN SERIN. Esta declaración sirve para recibir datos seriales en un formato standard asincrónico usando 8 bits de dato, sin paridad y 1 stop bit, (8N1), y para poder utilizarlo, debemos incluirlo igualmente que para el SEROUT la línea **INCLUDE** “modedefs.bas” al inicio del programa, su forma de utilizar es la siguiente:

Es importante saber que esta declaración detiene el programa esperando a que ingrese un dato, y solamente cuando haya recibido un dato, continúa con la siguiente línea de programa, para hacer que solo espere un determinado tiempo y luego continúe con la siguiente línea debe utilizar Timeout, para mayor información refiérase al manual de pbp, o utilice una interrupción externa para atender el dato a recibir, ver literal 5.10.1 Utilizando la interrupción en el puerto B.0.

El presente proyecto consiste en enviar un texto desde la ventana de comunicación serial de microcode hacia el PIC y este a su vez mostrará el texto en un LCD 2x16.

- a más de los anteriores
- 1 resistencia de 22 K Ω
- 1 LCD 2 x 16
- 1 resistencia de 10 Ω
- 1 potenciómetro de 10 K Ω



INCLUDE "modedefs.bas"	;incluyen los modos de comunicación
dat VAR BYTE	;variable de almacenamiento de 255
LCDOUT \$fe,1 , " LCD listo"	;texto para verificar la conexión
PAUSE 1000	;espera 1 seg. Con el texto "LCD listo"
LCDOUT \$fe,1	;limpia pantalla del Lcd y ubica cursor en casilla \$80
Inicio:	
SERIN portb.0,N2400,dat	;esperar el dato y guardarlo en dat
LCDOUT ,dat	;desplegar el dat en LCD
GOTO inicio	;ir a esperar el siguiente caracter
END	

Figura 5.9.5.2. Serin PC-PIC.bas Programa para recibir datos serialmente desde el PC a un PIC sin utilizar el CI. MAX232.

Una vez que arranca el PIC saldrá un texto que dice LCD listo, un segundo después se borrará y quedará en un lazo de espera del dato serial, si el texto inicial no sale, revise las conexiones al LCD, caso contrario si todo está bien, abrimos la ventana de comunicación serial de microcode de la forma que aprendimos anteriormente y escribimos en la ventana que dice Send: micro PIC, luego pulsamos la tecla Send Text, inmediatamente debemos ver el mismo texto en el LCD.

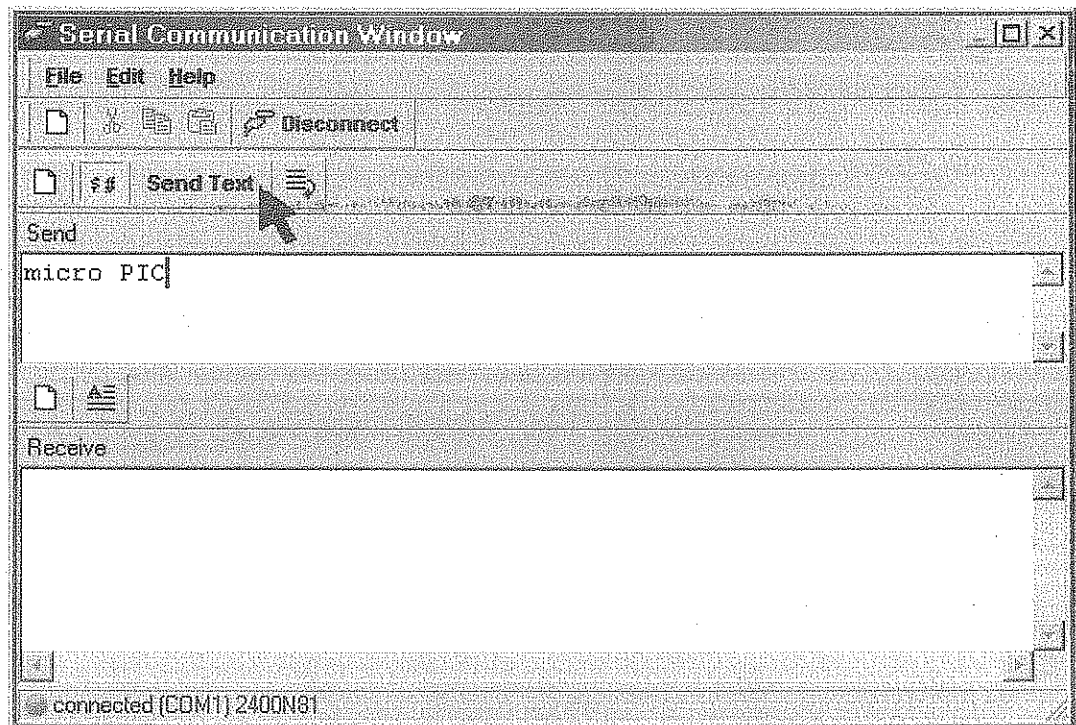


Figura 5.9.5.3. Pantalla de la ventana de comunicación serial listo para enviar un texto.

5.9.6. COMUNICACIÓN SERIAL CON EL CI. MAX 232.

El CI. MAX232 es la solución para transmitir a mayor distancia, ya que incrementa los niveles de voltaje de 5 V. a $\pm 10\text{V}$. gracias a un juego de capacitores que le ayuda a doblar los voltajes, por lo que para su alimentación solo requiere una fuente de 5V. que puede ser la misma que utiliza el PIC. El MAX232 dispone de 2 juegos de transmisores y receptores, de los cuales solo ocuparemos un par de ellos.

El MAX232 en este caso nos ayudará a convertir los voltajes TTL del PIC en voltajes de la norma RS232, quiere decir que si enviamos un estado lógico alto (5V.), a la salida del Tout del CI. MAX232 tendremos -10V. y si enviamos un 0 lógico desde el PIC (0V.), el MAX232 enviará +10V., por lo tanto debemos invertir el dato de la salida del PIC y esto lo conseguimos utilizando T2400 de la siguiente manera:

```
SEROUT portb.1,T2400,["D"]
```

;quiere decir enviar el dato serial D por el pin B.1 a
;2400bits/s 8N1 en dato verdadero, ver figura 5.9.3.1

El presente proyecto trabaja de la siguiente manera: una vez listo y conectado todo, cuando el PIC arranca debe encender el led y luego apagarlo, esto para asegurarnos que todo está funcionando bien, ahora desde el computador enviamos una letra cualquiera que no sea la C, observarán que el led parpadea cada que se le envía una letra, ahora si enviamos la C mayúscula el led se quedará encendido permanentemente e inmediatamente el PIC empezará a enviar un contador separado por el signo menos (-) empezando desde el 0 hasta el 255, como podemos observar esto es un ejemplo de la transmisión half-duplex.

MATERIALES

- 1 CI. MAX232
- 4 capacitores de 10 uF electrolíticos o preferible de tantalio
- de 2 a 30 mts de cable de 2 pares de hilos
- 1 conector DB9 hembra con su cajetín
- 1 led
- 1 resistencia de 330Ω.

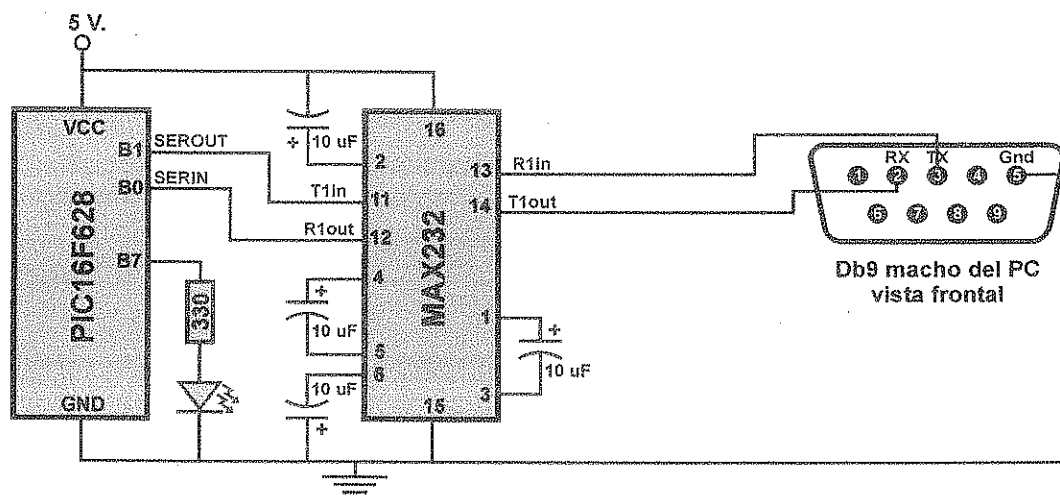


Figura 5.9.6.1. Diagrama de conexión del PIC y el CI. MAX232 para enviar y recibir datos entre un PC y el PIC.

INCLUDE "modedefs.bas"	;incluyen los modos de comunicación
led VAR portb.7	;nombre led al puerto b.7
dat VAR BYTE	;variable de almacenamiento de 255
num VAR BYTE	;variable para almacenar el contador
num =0	;contenido inicial para la variable num
GOSUB ledr	;ir a leds para saber si ya arrancó el PIC
Inicio:	
SERIN portb.0,T2400,dat	;esperar el dato y guardarlo en dat
IF dat = "C" THEN contar	;si dat es una C empiece a contar
GOSUB ledr	
GOTO inicio	;ir a esperar el siguiente caracter
contar:	
HIGH led	;led solo encendido indica enviando datos
SEROUT portb.1,T2400,[#num,"-"]	;enviar el contenido decimal de la variable num
	;seguido de un signo menos
num = num +1	;incrementar la variable 1 x 1
PAUSE 1000	;esperar 1 s
GOTO contar	;ir a subrutina contar
ledr:	;subrutina ledr
HIGH led	
PAUSE 200	
LOW led	
RETURN	
END	

Figura 5.9.6.2. TX-RX-MAX232.bas Programa para intercambiar datos entre una PC y un PIC utilizando el CI. MAX232.

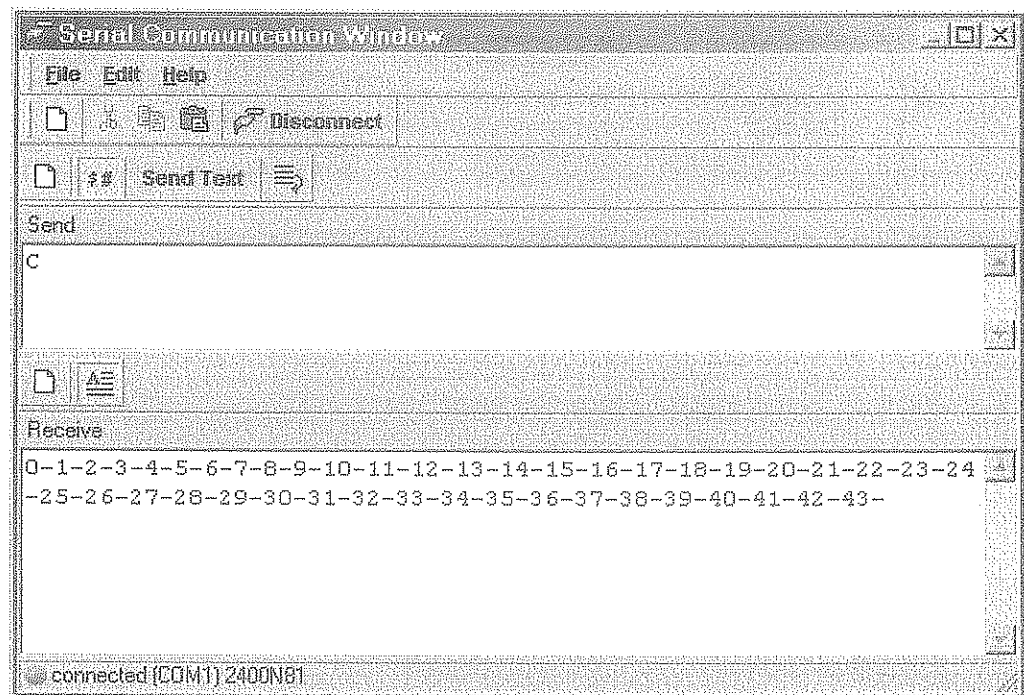


Figura 5.9.6.3. Pantalla de comunicación serial enviando y recibiendo datos desde el PIC.

5.9.7. COMUNICACIÓN SERIAL PIC A PIC.

Este proyecto consiste en hacer una transmisión simplex entre un PIC transmisor y un PIC receptor, el primero dispone de 3 botones, el botón A envía la letra "A" el cual el PIC receptor lo detecta y enciende un led rojo por 1 segundo, desde el transmisor presionamos el botón B y transmite la letra "B", el receptor encenderá una led amarillo, igualmente después de un segundo lo apagará y por último desde el transmisor presionamos la tecla C y el receptor encenderá un led verde. Por tratarse de una práctica y no complicarnos con el MAX232, ya que necesitaríamos 2, solo realizaremos a una distancia de 2 metros conectando directamente de PIC a PIC utilizando dato invertido (N2400).

MATERIALES.

- 2 mts de cable de 2 de hilos
- 3 leds un rojo, un verde y un amarillo
- 3 resistencias de $4,7\text{ K } \Omega$
- 3 resistencias de 330Ω
- 3 pulsadores NA
- 2 PIC16F628.

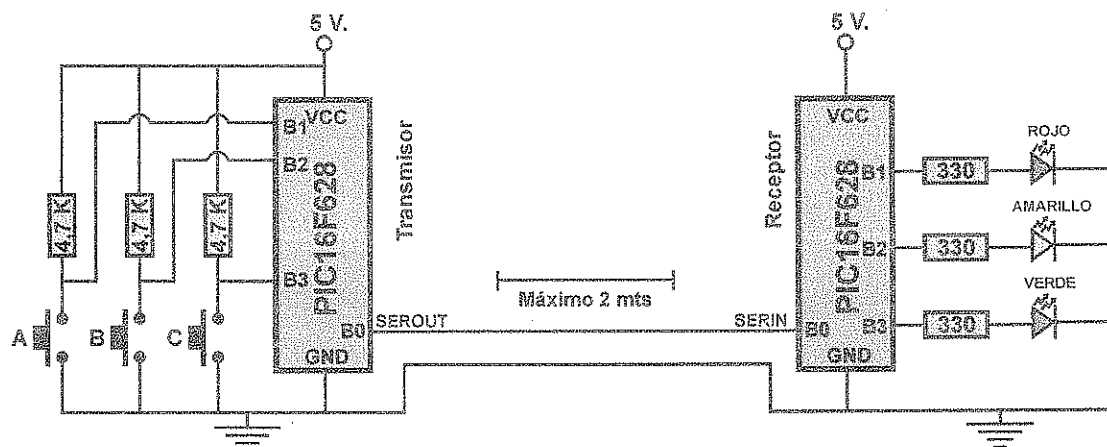


Figura 5.9.7.1. Diagrama de conexión para hacer una comunicación serial desde un PIC hacia otro PIC.

Para este proyecto podemos utilizar una o dos fuentes de 5 voltios, pero sería recomendable utilizar 2 fuentes para que la comunicación se vea más real, como es de suponerse se necesitará de 2 programas diferentes uno para cada microcontrolador, por lo que empezaremos con el programa del transmisor.

```
INCLUDE "modedefs.bas"
botonA VAR portb.1
botonB VAR portb.2
botonC VAR portb.3
```

```
;incluyen los modos de comunicación
;nombre botonA al puerto b.1
```

continúa ...

```

transmitir:
IF botonA=0 THEN envio1           ;si el botonA es presionado ir a envio1
IF botonB=0 THEN envio2           ;si el botonB es presionado ir a envio2
IF botonC=0 THEN envio3           ;si el botonC es presionado ir a envio3
GOTO transmitir

envio1:
SEROUT portb.0,N2400,["A"]         ;enviar "A" a 24008N1
PAUSE 500
GOTO transmitir

envio2:
SEROUT portb.0,N2400,["B"]         ;enviar "B" a 24008N1
PAUSE 500
GOTO transmitir

envio3:
SEROUT portb.0,N2400,["C"]         ;enviar "C" a 24008N1
PAUSE 500
GOTO transmitir

END

```

Figura 5.9.7.2. transmisor PIC.bas Programa para enviar datos desde un PIC.

```

INCLUDE "modedefs.bas"             ;incluyen los modos de comunicación

ledr VAR portb.1                   ;nombre ledr al puerto b.1
leda VAR portb.2
ledv VAR portb.3
datos VAR BYTE                     ;variable para almacenar el dato serial

HIGH ledr                          ;led para saber si ya arrancó el PIC
PAUSE 500
LOW ledr

recibir:
SERIN portb.0,N2400,datos           ;recibir dato serial y guardar en datos
IF datos="A" THEN HIGH ledr:PAUSE 1000 ;si es A encender ledr y esperar 1 seg.
IF datos="B" THEN HIGH leda:PAUSE 1000
IF datos="C" THEN HIGH ledv:PAUSE 1000

LOW ledr:LOW leda:LOW ledv
GOTO recibir

END

```

Figura 5.9.7.3. Receptor PIC.bas Programa para recibir datos desde un PIC.

Una vez que concluya con esta práctica pruebe con el MAX232 a mayor distancia o el receptor cambie los leds por un LCD y escriba desde el transmisor agregando un teclado de 16 pulsadores.

5.9.8. COMUNICACIÓN SERIAL RS422/485.

Para entender que es lo que vamos a hacer explicaremos en pocas palabras que es la interfaz serial RS422 y que es la interfaz serial RS485, estas se diseñaron para la conexión física entre computadores y terminales directamente. Estos estándares tienen grandes ventajas con respecto a la norma RS232 como por ejemplo, la distancia de comunicación de hasta 1200 mts, la velocidad de transmisión de hasta 10 Mbits/seg. y el número de elementos a conectarse, para la interfaz RS422 pueden conectarse un transmisor y hasta 10 receptores en un modo de transmisión Full-duplex, mientras que para la interfaz RS485 se pueden conectar simultáneamente hasta 32 transmisores/receptores en un sistema half-duplex, otra ventaja frente al sistema RS232 es que no requiere fuentes duales sino una fuente de alimentación de 5 voltios.

Ambas interfaces utilizan el método de medida diferencial, en la que utilizan dos líneas para la transmisión y dos para recepción, en cada par de conductores la segunda tiene un nivel de voltaje complementario al del primero, y el receptor responde a la diferencia de voltajes entre los dos conductores. Este tipo de líneas de transmisión se llaman balanceadas, y esto permite la eliminación de ruidos electrostáticos y electromagnéticos común en las dos líneas que se utilizan.

El CI. 75176 contiene un transmisor y un receptor y solo dos líneas diferenciales A y B de entrada/salida de datos, dos líneas adicionales RE y DE determinan la función que debe cumplir el integrado, permitiendo o inhibiendo la recepción o la transmisión de datos. Para este integrado el transmisor es habilitado por un 1 lógico y un 0 lógico habilita el receptor, estas dos líneas RE y DE son unidas a un puerto del PIC, en donde el microcontrolador determinará cuando transmitir y cuando recibir datos, en un sistema half-duplex.

El sistema RS422 establece una comunicación full-duplex para lo cual se requiere 2 líneas adicionales, esto se lo consigue agregando otro CI. 75176 exclusivamente para la transmisión por lo que RE y DE estarán conectados en nivel lógico alto, mientras el otro CI. 75176 se lo configura para recepción conectando los terminales control a un nivel 0 lógico, de esta manera quedan operando los 2 circuitos lineales 75176, con la ventaja de poder transmitir y recibir al mismo tiempo.

Como el microcontrolador PIC16F628 ejecuta línea por línea del programa, no es capaz de leer y recibir un dato a la vez, por lo tanto no se puede realizar una comunicación full-duplex, de esta manera no es aplicable una comunicación con la interfaz RS422, lo ideal es la interfaz RS485 ya que está diseñado para una transmisión half-duplex y de hecho este es el modo de transmisión que se utiliza en la mayoría de dispositivos comerciales basados en microcontroladores, un ejemplo de ello es la comunicación del teclado con la central de alarma, esta lo realiza mediante interfaz RS485 de 3 hilos (A, B y masa) y un cuarto hilo alimenta de 5V. al teclado. Una recomendación importante es que el cable a utilizarse debe ser del tipo par trenzado (Twisted-Pair), que consiste en dos conductores aislados retorcidos entre sí con lo cual se consigue una mayor inmunidad al ruido electromagnético, si el cable tiene adicionalmente una hoja conductora (blindaje) rodeándolo, se obtiene más inmunidad.

5.9.9. COMUNICACIÓN SERIAL PIC A PIC CON LA INTERFAZ RS485.

El presente proyecto consiste hacer una transmisión recepción entre dos PIC'S con una interfaz RS485 y en modo de transmisión full-duplex comúnmente visto en centrales de alarmas.

Su funcionamiento consta de la siguiente manera: al arrancar los micros ambos hacen encender los leds rojos por un instante, esto nos indica que empezaron a funcionar, el transmisor consta de 2 pulsadores y un led, y el receptor de 2 leds, un rojo un verde y un pulsador, cuando presionamos el pulsador A del transmisor enciende el led rojo y presionando el pulsador B enciende el led verde, bien hasta aquí hemos realizado una comunicación de un solo sentido,

ahora lo intercambiamos para hacer que el PIC receptor se convierta en transmisor y lo mismo con el otro PIC de transmisor a receptor, para esto en el PIC transmisor presionamos las 2 teclas al mismo tiempo por un segundo, luego de esto notarán que cada tecla ya no tiene efecto como lo hacia antes, es decir encendía cada uno de los 2 leds del receptor, ahora vamos al receptor el cual ahora es transmisor y presionamos el único pulsador que dispone, notarán que el led del que antes era transmisor parpadea 2 veces, para volver al modo que iniciaron solo debemos apagar y volver aprender ambos micros.

MATERIALES.

- cable de 2 pares de hilos, preferible del tipo par trenzado
- 3 leds dos rojos y un verde
- 3 resistencias de 4,7 K Ω
- 3 resistencia de 330 Ω
- 3 pulsadores NA
- 2 PIC16F628
- 2 circuitos lineales 75176.

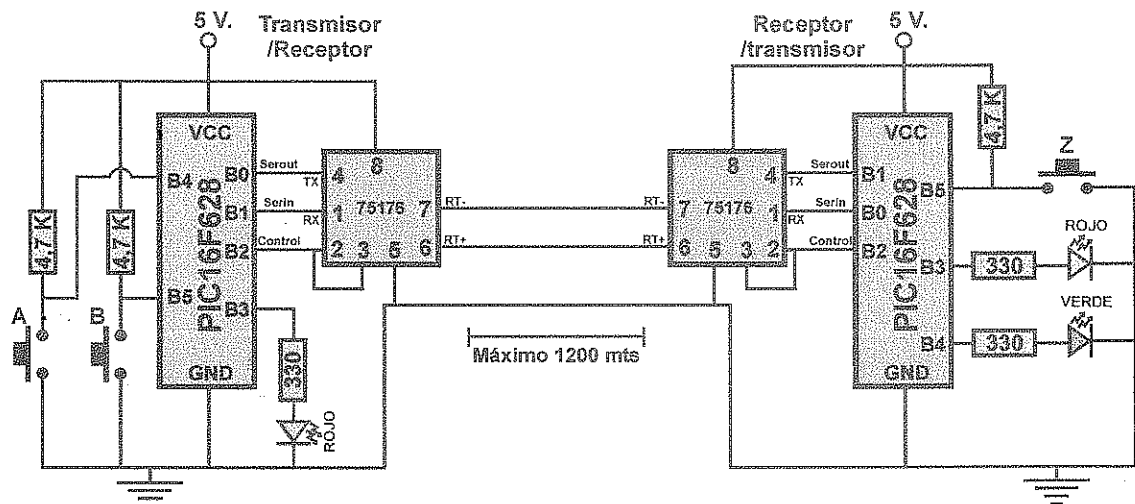


Figura 5.9.9.1. Diagrama de conexión para hacer una comunicación serial desde un PIC hacia otro PIC con interfaz serial RS485.

```
INCLUDE "modedefs.bas"
```

```
control VAR portb.2
```

```
led VAR portb.3
```

```
recib VAR BYTE
```

```
GOSUB rojo
```

```
envio:
```

```
HIGH control
```

```
IF (portb.4=0) AND (portb.5=0) THEN receptor
```

```
IF portb.4=0 THEN ledrojo
```

```
;incluyen los modos de comunicación
```

```
;pin para control TX RX
```

```
;nombre para pin b.3
```

```
;variable para almacenar dato
```

```
;led para saber si ya arrancó el PIC
```

```
;subrutina envio
```

```
;control modo transmisor TX
```

```
;si los 2 botones están presionados
```

```
continúa ...
```

```

IF portb.5=0 THEN ledverde
GOTO envio

ledrojo:                                ;subrutina ledrojo
    GOSUB rojo                          ;encender el led
    SEROUT portb.0,T2400,["A"]          ;enviar la A por puerto b.0
    PAUSE 500
    GOTO envio                          ;ir nuevamente a envío

ledverde:
    GOSUB rojo                          ;encender el led
    SEROUT portb.0,T2400,["B"]          ;enviar la B por puerto b.0
    PAUSE 500
    GOTO envio                          ;ir nuevamente a envío

receptor:                                ;subrutina receptor
    GOSUB rojo                          ;encender el led
    SEROUT portb.0,T2400,["C"]          ;enviar la C
    PAUSE 500

recibiendo:                             ;subrutina recibiendo
    LOW control                         ;cambiar a modo receptor RX
    SERIN portb.1,T2400,recib           ;esperar dato y guardarlo en recib
    IF recib="Z" THEN perfecto          ;si recib es una Z ir a perfecto
    GOTO recibiendo                    ;caso contrario seguir en recibiendo

rojo:
    HIGH led                           ;encender led rojo
    PAUSE 200                          ;esperar 200 mls
    LOW led                             ;apagar led rojo
    PAUSE 200
    RETURN                              ;retornar al gosub que lo envió

perfecto:
    GOSUB rojo                          ;encender 2 veces el led rojo
    GOSUB rojo
    GOTO recibiendo                    ;ir a recibiendo

END

```

Figura 5.9.9.2. transmisor-RS485.bas Programa para enviar datos PIC a PIC en interfaz RS485.

```

INCLUDE "modedefs.bas"                  ;incluye los modos de transmisión
control VAR portb.2                     ;pin para el control
ledr VAR portb.3
ledv VAR portb.4
date VAR BYTE                           ;variable para almacenar dato serial

GOSUB rojo                              ;led para saber si ya arrancó el PIC

```

continúa

recibir:	;subrutina recibir
LOW control	;control modo receptor
SERIN portb.0,T2400,date	;esperar por dato serial y guardar
IF date="A" THEN ok1	;si dato es una A ir a ok1
IF date="B" THEN ok2	
IF date="C" THEN transmisor	
HIGH ledv : HIGH ledr	;se enciende los 2 leds cuando el
PAUSE 2000	;dato recibido no es A,B, ni C
LOW ledv :LOW ledr	
PAUSE 500	
GOTO recibir	; ir a recibir
ok1:	;subrutina ok1
GOSUB rojo	;ir y retornar de rojo
GOTO recibir	
ok2:	;subrutina ok2
HIGH ledv	;encender el led verde
PAUSE 200	
LOW ledv	
GOTO recibir	
transmisor:	;subrutina modo transmisor
HIGH control	;control cambia a transmisor
IF portb.5=0 THEN	;si presionamos el botón del portb.5 entonces
SEROUT portb.1,T2400,["Z"]	;enviar la Z por puerto b.1
PAUSE 500	;esperar 0,5 seg.
ENDIF	;fin de la condición
GOTO transmisor	;volver a transmisor
rojo:	;subrutina rojo
HIGH ledr	;encender el led rojo
PAUSE 200	
LOW ledr	
PAUSE 200	
RETURN	; retornar al gosub que lo envió
END	

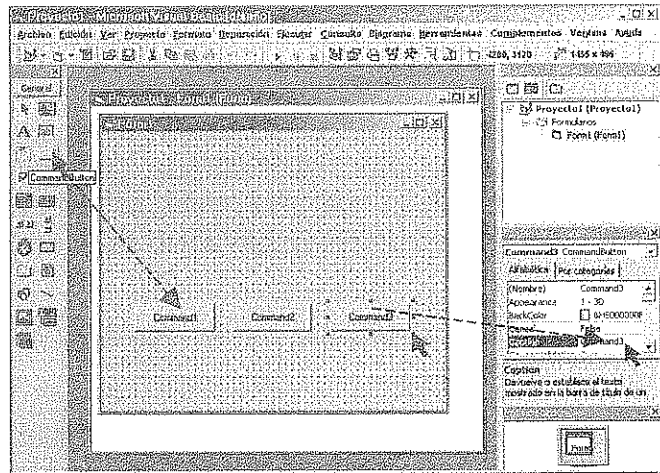
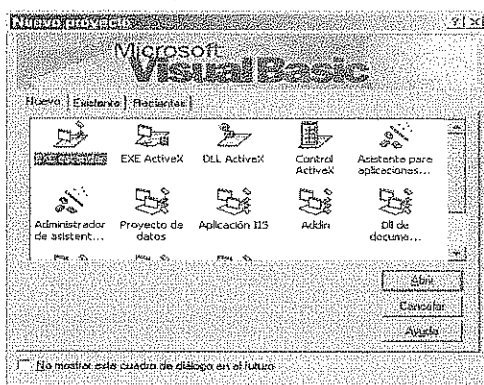
Figura 5.9.9.3. receptor-RS485.bas Programa para recibir datos PIC a PIC en interfaz RS485.

NOTA: Si la comunicación es errónea o está desconectado el transmisor, el receptor encenderá los 2 leds a la vez, el rojo y el verde durante 2 segundos y luego lo apagará por 0,5 seg.

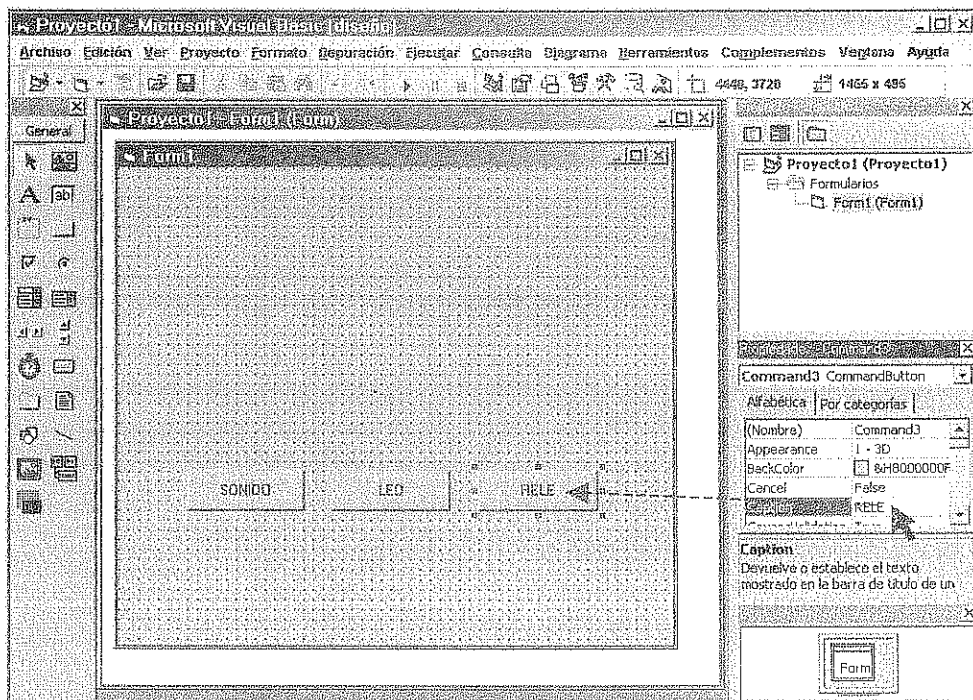
5.9.10. COMUNICACIÓN SERIAL DE VISUAL BASIC Y PIC.

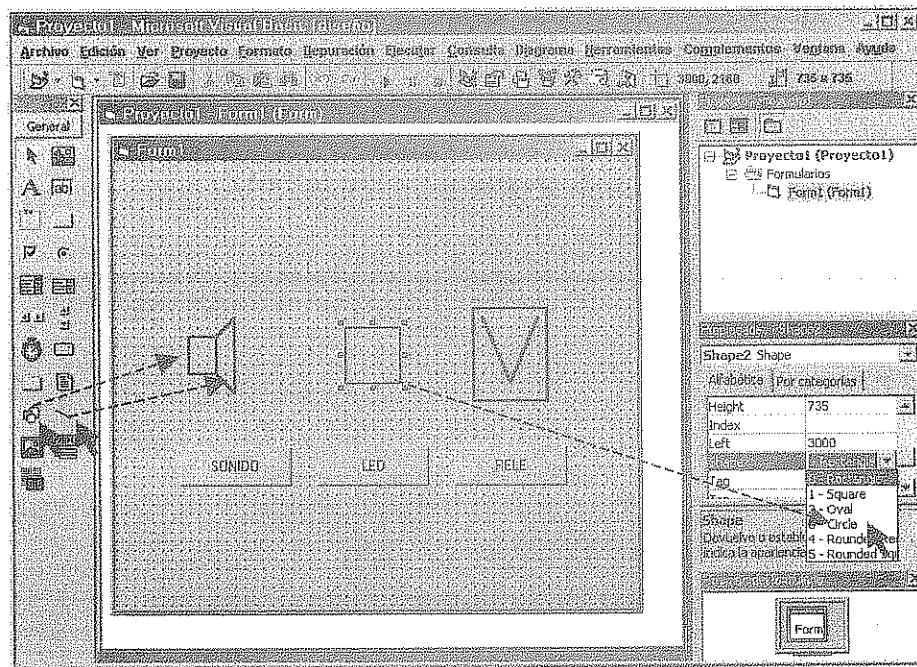
Esta es la parte de la electrónica que más entusiasmo genera, poder controlar a través de un computador todos los periféricos de un edificio (ascensor, luces, alarmas, cerraduras, etc), controlar los movimientos de un brazo robótico, controlar la producción de una fábrica (motores,

bombas, calefactores, etc), todo desde la pantalla de un computador. ¿Les parece interesante?, pues como para introducimos en el mundo del control computarizado haremos un control de un relé, una chicharra y un led, los cuales nos responderán si están activados o no, para hacer el tablero de control se necesita saber programar en VISUAL BASIC 6.0, de todas maneras daremos unas indicaciones para poder crear un tablero básico. Empezaremos por diseñar los botones en un form de VB, para esto ejecutamos el programa VB, en la pantalla principal escogemos exe estándar y damos clic en Abrir.

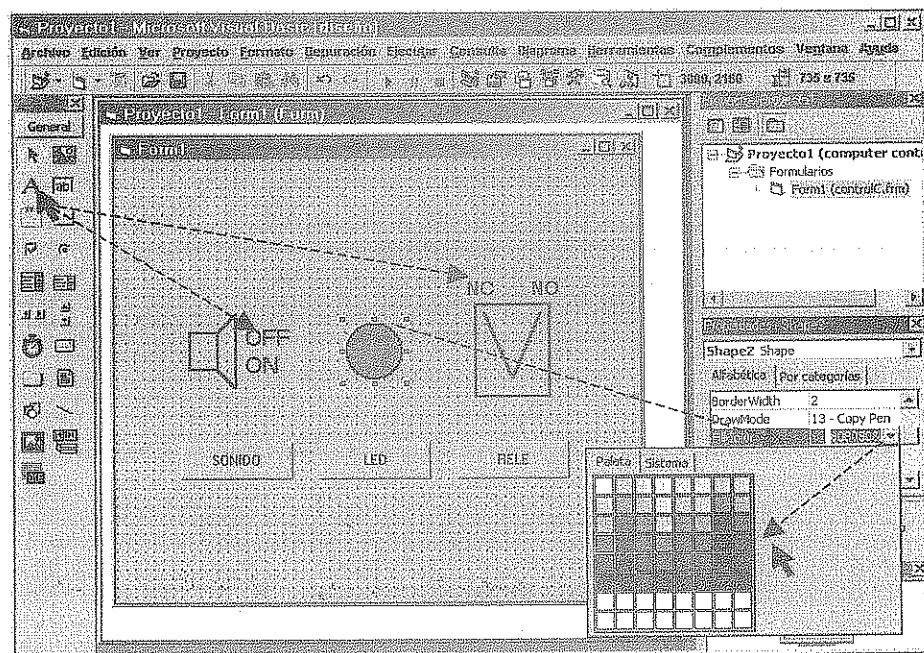


Como pueden ver en la pantalla de la derecha hemos creado 3 botones con la herramienta commandButton, si desean pueden cambiar el texto de command3 por ejemplo por RELE, para esto primero debemos seleccionar el objeto en este caso command3 y en propiedades del objeto que se encuentra al lado derecho específicamente en **Caption** **Command3**, lo borramos y escribimos **RELE**, en el otro botón escribimos **LED** y en el último **SONIDO**, quedará como ilustra el siguiente gráfico:



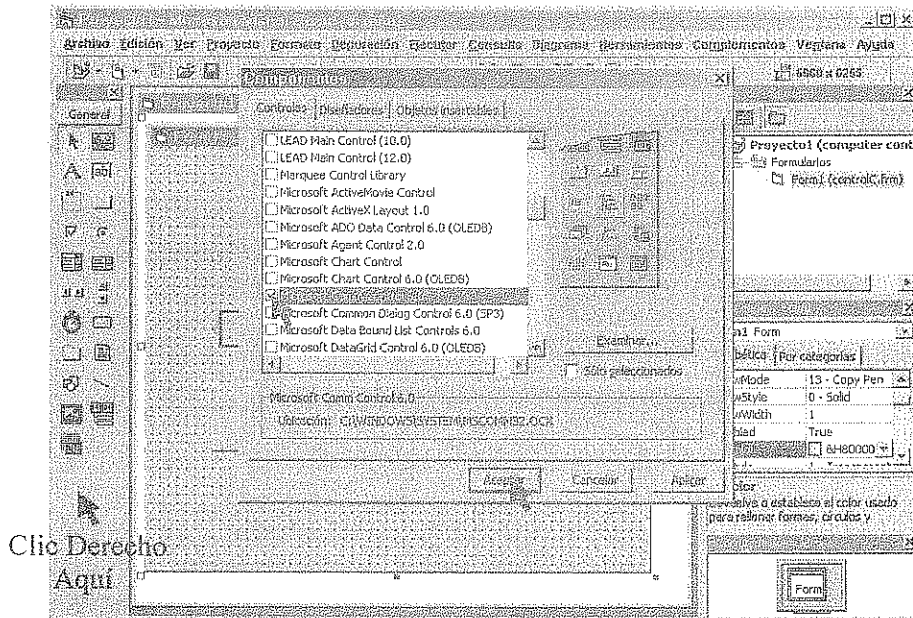


Ahora como ven arriba hemos dibujado un parlante y un relé, explicamos como se hizo cada uno, primero escogemos la herramienta del lado izquierdo **Shape**, que sirve para dibujar rectángulos círculos óvalos, en este caso dibujamos el parlante, hacemos un rectángulo y luego cogemos la herramienta **Line** y dibujamos las líneas de la bocina, si queremos que las líneas sean más gruesas, en el lado derecho en propiedades del objeto **BorderWidth** [1], lo colocamos el [2] y esto hará más gruesas las líneas que seleccionemos. De esta misma forma creamos el relé y ponemos 2 líneas para indicar el cambio de estado del relé. Para crear el LED utilizamos **Shape** mismo y primero hacemos un cuadrado, luego vamos al lado derecho en propiedades en **Shape** debe decir [0 rectangle], lo ponemos [3 circle] y se convertirá en un círculo, cambiamos el grosor a 2 y listo.

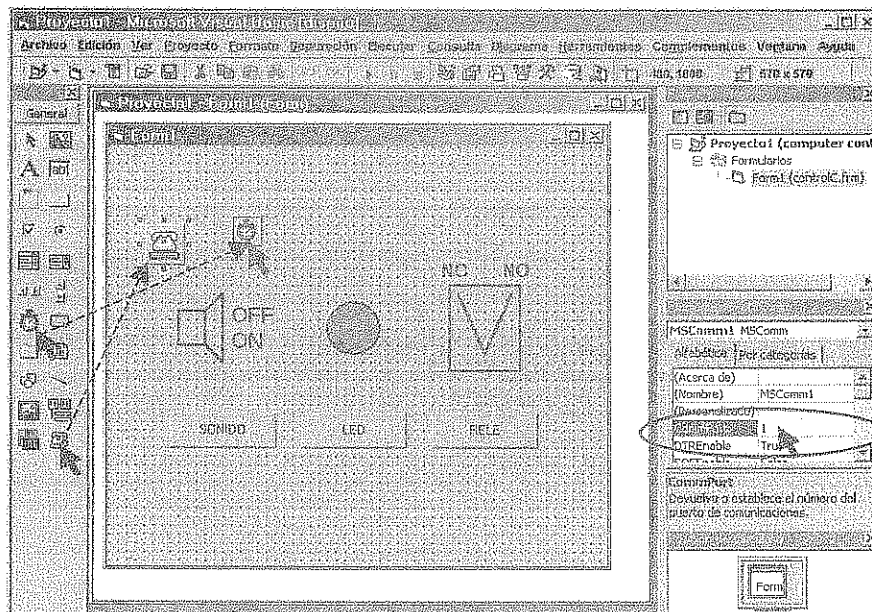


Para crear textos presionamos en el lado izquierdo en label, lo colocamos en el lugar que deseamos y luego en **Caption** ponemos ON, OFF, NC, NO. Para cambiar el estilo de letra y el tamaño, primero seleccionamos el texto a modificar y en el lado derecho en propiedades **Font**, elegimos los cambios que deseemos y listo.

Para dar color al LED, seleccionamos el círculo y en propiedades donde dice **FillStyle** **transparent** cambiamos a **Solid**, luego en **FillColor** escogemos la paleta y ponemos el color plomo, para indicar que el led está apagado.



Para habilitar la comunicación serial, damos un clic con el botón derecho sobre el cuadro General y escogemos la opción componentes, luego saldrá una pantalla con una lista de componentes y buscamos **Microsoft Comm Control 6.0**, seleccionamos y damos clic en Aceptar, notaremos que ahora aparece un icono nuevo *un teléfono*, colocamos este teléfono en la form, y en las propiedades **Commport** podemos modificar si es com1 o com2, también modificamos la velocidad de transmisión que por defecto viene con 9600,n,8,1, y también colocamos un Timer.



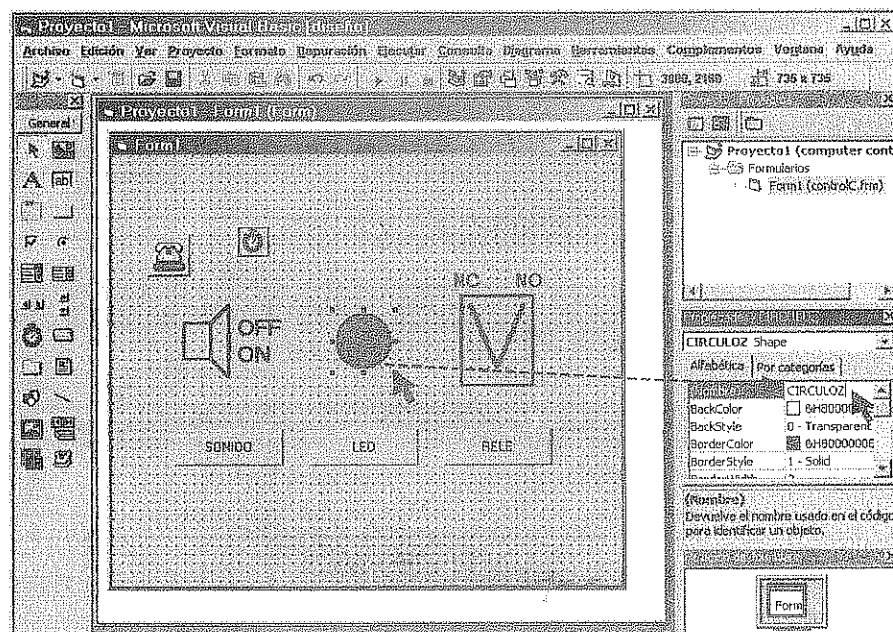
Bien ahora la palabra ON y la línea del relé que señala al NO, no deben aparecer, sino hasta cuando el microcontrolador se los indique, por tal razón debemos ocultarlos, y esto lo realizamos seleccionando a cada uno y en propiedades donde dice **Visible** **True**, lo cambiamos por **False**, bien en ese instante no desaparecerán sino hasta cuando se ejecute el programa.

Es importante darles nombres a cada objeto solo a los que necesariamente vamos a modificar, estos se lo realiza de la siguiente manera, primero seleccionamos, digamos que el botón SONIDO, al lado derecho el primer item de propiedades dice **(Nombre)** **Command1** aquí lo ponemos el mismo nombre del objeto con una Z al final así **SONIDOZ**, esto lo hacemos con el objeto de no confundir un nombre Caption SONIDO con el nombre del objeto SONIDOZ, lo mismo hacemos con los siguientes objetos: la palabra ON y OFF, las tres teclas SONIDO, LED y RELE, la línea del relé que señala a la palabra NO y la que señala a NC también, todos ellos su nuevo nombre a cambio de: Command1, Command2, Command3, ShapeX, LabelX, LineX, serán:

Propiedad	N. antiguo	Propiedad	N. nuevo
(Nombre)	CommandX	(Nombre)	SONIDOZ
(Nombre)	CommandX	(Nombre)	LEDZ
(Nombre)	CommandX	(Nombre)	RELEZ
(Nombre)	ShapeX	(Nombre)	CIRCULOZ
(Nombre)	LabelX	(Nombre)	OFFZ
(Nombre)	LabelX	(Nombre)	ONZ
(Nombre)	LineX	(Nombre)	LINEANOZ
(Nombre)	LineX	(Nombre)	LINEANCZ

Figura 5.9.10.1. Tabla que muestra los cambios de nombre a realizar.

A continuación una imagen que muestra como se cambia el nombre de un objeto, observen que se seleccionó el Círculo y en el lado derecho decía **(Nombre)** Shape2, se lo cambió por **(Nombre)** CIRCULOZ, de esta manera cambiamos a los 8 objetos ya indicados anteriormente con los nombres que aparece en la tabla de la figura 5.9.10.1.



Es hora de programar las funciones de los botones, para esto primero damos doble clic en cualquier parte de la form, esto hará que se abra otra pantalla (Código), en la que sale un texto así:

```
Private Sub Form_Load ( )
```

```
End Sub
```

Aquí escribimos habilitar el puerto y correr el Timer con intervalos de 1 mls, y empezamos a programar cada uno de los botones.

```
Private Sub Form_Load()
```

```
    MSComm1.PortOpen = True
```

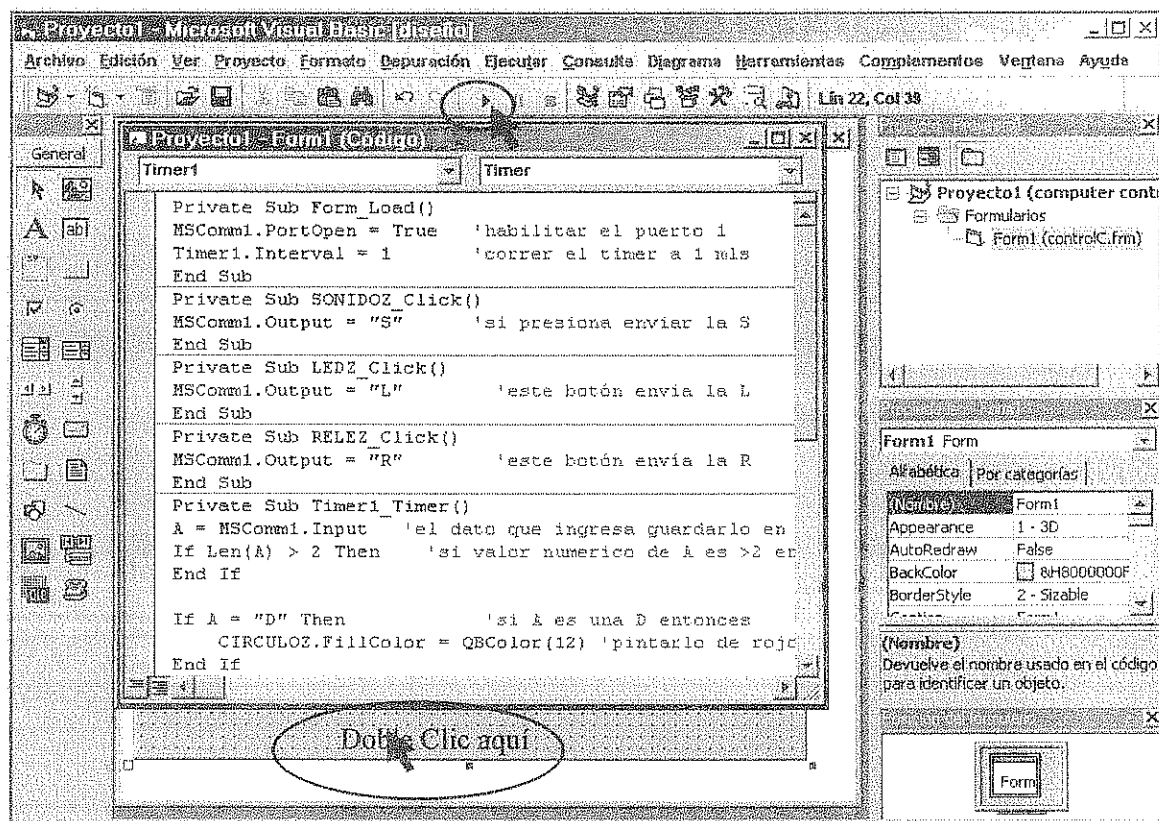
'habilitar el puerto comm1

```
    Timer1.Interval = 1
```

'correr el timer con intervalos de 1 mls

```
End Sub
```

Esto quiere decir que cuando se ejecute este programa corra su contenido, es decir abra el puerto com y empiece a correr el Timer con intervalos de 1 mls.



Si se quiere hacer comunicación solo desde el PC al PIC y no desde el PIC al PC, borramos las líneas desde donde dice: Private Sub Timer1_Timer() hasta el final del programa, también debemos borrar la línea que dice Timer1.Interval=1. y en la form borrar el Timer, y dejarlo el teléfono, ya que sí lo necesitamos para enviar datos desde el PC al PIC.

El programa completo quedaría así:

Private Sub Form_Load()		
MSComm1.PortOpen = True		'habilitar el puerto 1
Timer1.Interval = 1		'correr el timer a 1 ms
End Sub		
Private Sub SONIDOZ_Click()		
MSComm1.Output = "S"		'si presiona SONIDOZ SONIDO enviar la S
End Sub		
Private Sub LEDZ_Click()		
MSComm1.Output = "L"		'si presiona el botón LEDZ LED enviar la L
End Sub		
Private Sub RELEZ_Click()		
MSComm1.Output = "R"		'si presiona RELEZ RELE enviar la R
End Sub		
Private Sub Timer1_Timer()		
A = MSComm1.Input		'el dato que ingresa guardarlo en variable A
If Len(A) > 2 Then		'si el valor numérico de A es >2 entonces
End If		
If A = "D" Then		'si A es una D entonces
CIRCULOZ.FillColor = QBColor (12)		'pintar el círculo de rojo
End If		
If A = "E" Then		'si A es una E entonces
CIRCULOZ.FillColor = QBColor (8)		'pintar el círculo de plomo
End If		
If A = "G" Then		
LINEANCZ.Visible = False		'ocultar línea NC rele
LINEANOZ.Visible = True		'mostrar la línea NO relé
End If		
If A = "F" Then		
LINEANCZ.Visible = True		'mostrar la línea NC
LINEANOZ.Visible = False		'ocultar la línea NO
End If		
If A = "H" Then		
ONZ.Visible = True		'mostrar la palabra ON
OFFZ.Visible = False		'ocultar la palabra OFF
End If		
If A = "I" Then		
ONZ.Visible = False		'ocultar la palabra ON
OFFZ.Visible = True		'mostrar la palabra OFF
End If		
End Sub		

Figura 5.9.10.1. Programa para recibir y enviar datos desde el VISUAL BASIC 6.0 a un PIC.

NOTA: todo lo que está con fondo gris, sería el programa necesario para únicamente transmitir desde el PC al PIC, lo demás podemos borrarlo, junto con el icono de la form (el Timer).

INCLUDE "modedefs.bas"	;incluyen los modos de comunicación
serial VAR BYTE	;variable de almacenamiento de 255
sec1 VAR BIT	; variable sec1 de 1 bit 0 o 1
sec2 VAR BIT	; variable sec2 de 1 bit 0 o 1
sec3 VAR BIT	; variable sec3 de 1 bit 0 o 1
sec1=0	;valores iniciales para las variables
sec2=0	
sec3=0	
led VAR portb.5	;nombre para los pines
chicharra VAR portb.6	
rele VAR portb.7	
HIGH led:PAUSE 500:LOW led	;led para saber si ya arrancó el PIC
Inicio:	
SERIN portb.0,N9600,serial	;esperar por dato serial y guardarlo
IF serial="S" THEN sonidos	;si el dato es una S ir a sonidos
IF serial="L" THEN leds	;si el dato es una L ir a leds
IF serial="R" THEN rele	;si el dato es una S ir a sonido
GOTO inicio	
sonidos:	
IF sec1=0 THEN	;bandera para la chicharra 1 On y 0 es OFF
HIGH chicharra	
SEROUT portb.1,N9600,["H"]	;enviar H diciendo que esta prendido
sec1=1	
GOTO inicio	
ENDIF	
IF sec1=1 THEN	
LOW chicharra	
SEROUT portb.1,N9600,["I"]	;enviar I diciendo que esta OFF
sec1=0	
ENDIF	
GOTO inicio	
leds:	
IF sec2=0 THEN	
HIGH led	
SEROUT portb.1,N9600,["D"]	;envía D diciendo que el led es ON
sec2=1	
GOTO inicio	
ENDIF	
IF sec2=1 THEN	
LOW led	
SEROUT portb.1,N9600,["E"]	;envía E diciendo que el led es OFF
sec2=0	
ENDIF	
GOTO inicio	

continúa ...

```

reles:
  IF sec3=0 THEN
    HIGH rele
    SEROUT portb.1,N9600,["G"]           ;envía G diciendo que el relé es ON
    sec3=1
    GOTO inicio
  ENDIF
  IF sec3=1 THEN
    LOW rele
    SEROUT portb.1,N9600,["F"]           ;envía F diciendo que el relé es OFF
    sec3=0
  ENDIF
  GOTO inicio
END

```

Figura 5.9.10.3. Control-Visual-Basic.bas Programa para enviar y recibir datos desde el PIC.

Como para probar si los datos que salen del PIC son correctos, puede conectar a la ventana de comunicación serial de microcode, a **9600 baudrate**, y en Send Text, envíe las letras mayúsculas una por una la S, S, L, L, R, R, y de respuestas debe recibir lo siguiente: HIDEGF.

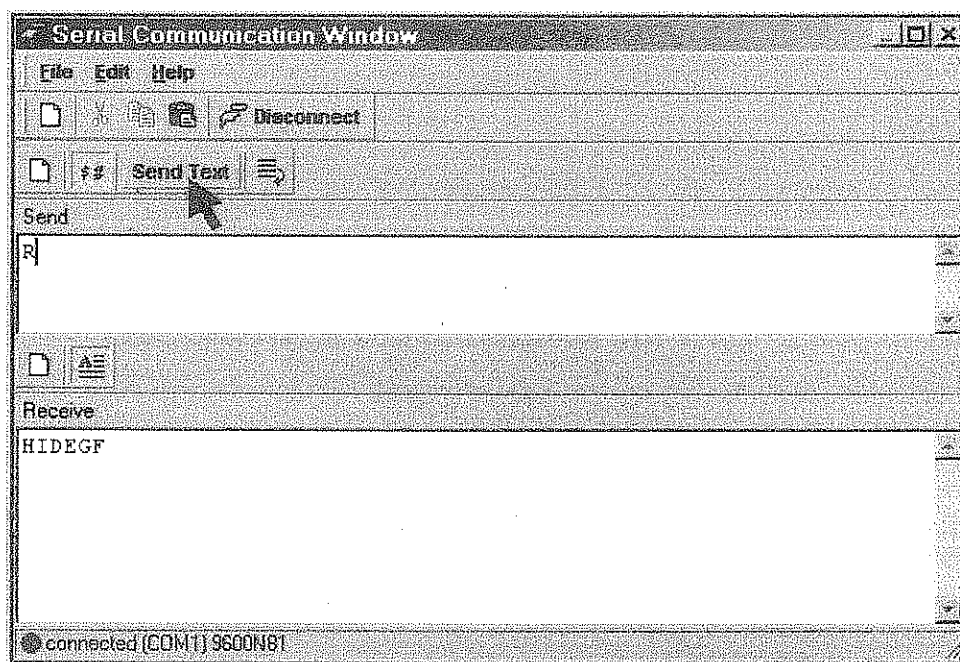


Figura 5.9.10.4. Pantalla de comunicación serial con lo datos de respuestas si está activado la chicharra(H) o no(I), encendido el led (D) o no(E), conectado el relé (G) o no (F).

La comunicación con el programa hecho en Visual Basic es similar, con la diferencia que esas mismas letras activa la palabra ON, cambia de color el círculo, y activa la línea NO.

5.9.11. COMUNICACIÓN SERIAL SINCÓNICA I²C.

Muchos de los dispositivos electrónicos que se encuentran comúnmente en una tarjeta electrónica, incluyen circuitos integrados con el bus I²C, como por ejemplo las memorias 24CXX, los procesadores de señal, codificadores de video, sensores de temperatura, RTC (reloj en tiempo real), sensores ultrasónicos, etc.

El bus I²C (Inter Integrated Circuit), necesita solo 2 líneas para transmitir y recibir datos, estos son: para datos (SDA) y para la señal de reloj (SCL), esta forma de comunicación utiliza una sincronía con un tren de pulsos que viaja en la línea SCL, de tal manera que en los flancos negativos se revisan los datos RX o TX. Ver figura 5.9.12.2., su velocidad de transmisión pueden ser de 100Kbits/seg. en el modo standard, 400Kbits/seg. en el modo rápido y 3,4Mbits/seg. en alta velocidad. Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software, por lo que existe una relación permanente Master/Slave. El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de reloj (SCL), y el Slave es el dispositivo direccionado, sin embargo cada dispositivo reconocido por su código (dirección), puede operar como transmisor o receptor de datos, ya que la línea (SDA) es bidireccional.

5.9.12. COMUNICACIÓN I²C CON UNA MEMORIA SERIAL 24LC04B.

Esta es una práctica muy básica para aprender sobre la interfaz I²C, consiste en guardar datos en las cuatro primeras direcciones de la memoria serial, estas son utilizadas para el almacenamiento de datos que pueden ser necesitados más adelante. Para el caso de la memoria 24LC04B tiene un espacio de memoria de 4Kbytes, luego de almacenarlos los volveremos a leer y mostrar en la pantalla de un LCD.

Referencia	Capacidad	Ciclos de E/W	Bloques internos	Dirección			Cantidad de disposit. en el bus	Voltaje de operación
				A0	A1	A2		
24LC01B	1K bits	1,000.000	1	1-0	1-0	1-0	8	2,5-5,5V.
24LC02B	2K bits	1,000.000	1	1-0	1-0	1-0	8	2,5-5,5V.
24LC04B	4K bits	1,000.000	2	X	1-0	1-0	4	2,5-5,5V.
24LC08B	8K bits	1,000.000	4	X	X	1-0	2	2,5-5,5V.
24LC016B	16K bits	1,000.000	8	X	X	X	1	2,5-5,5V.

Figura 5.9.12.1. Tabla de las capacidades de las memorias EEPROM y su direccionamiento, en nuestro caso solo podemos poner hasta 4 memorias seriales en una red I²C.

El principio de funcionamiento es el siguiente: primero se envía el star bit (bit de arranque) cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida debe contener la dirección del esclavo seleccionado, en este caso se envía el código de la memoria 1010 (este dato lo suministra el fabricante), luego la dirección del dispositivo (A2, A1, A0), y un bit 0 indicando que se desea escribir en la memoria (1=lectura), luego de todo esto la memoria debe enviar un reconocimiento para informarle al microcontrolador que recibió la información, este acuse de recibo se denomina ACK (acknowledge).

Luego el Master lee el ACK, si vale 0 (enviado por el esclavo), el proceso de transferencia continúa. Si vale 1, esto indica que el circuito direccionado no valida la

comunicación, entonces el Maestro genera un bit de stop para liberar el bus I²C, en la cual las líneas SDA y SCL pasan a un estado alto, vamos a suponer que el ACK es 0, entonces el microcontrolador envía los 8 bits correspondientes a la posición de memoria que se desea escribir o leer, nuevamente la memoria envía un reconocimiento, finalmente se envía el dato a ser almacenado y se espera la respuesta de la memoria indicando que el dato llegó correctamente, finalmente se debe enviar el bit de parada.

Como en nuestra práctica vamos a almacenar la palabra HOLA, vamos a suponer que ya guardamos las 3 primeras letras, la H en la dirección 0, la O en la dirección 1 y la L en la dirección 2, nos falta guardar la A en la dirección 3, esto se realizaría enviando pulsos de la siguiente manera:

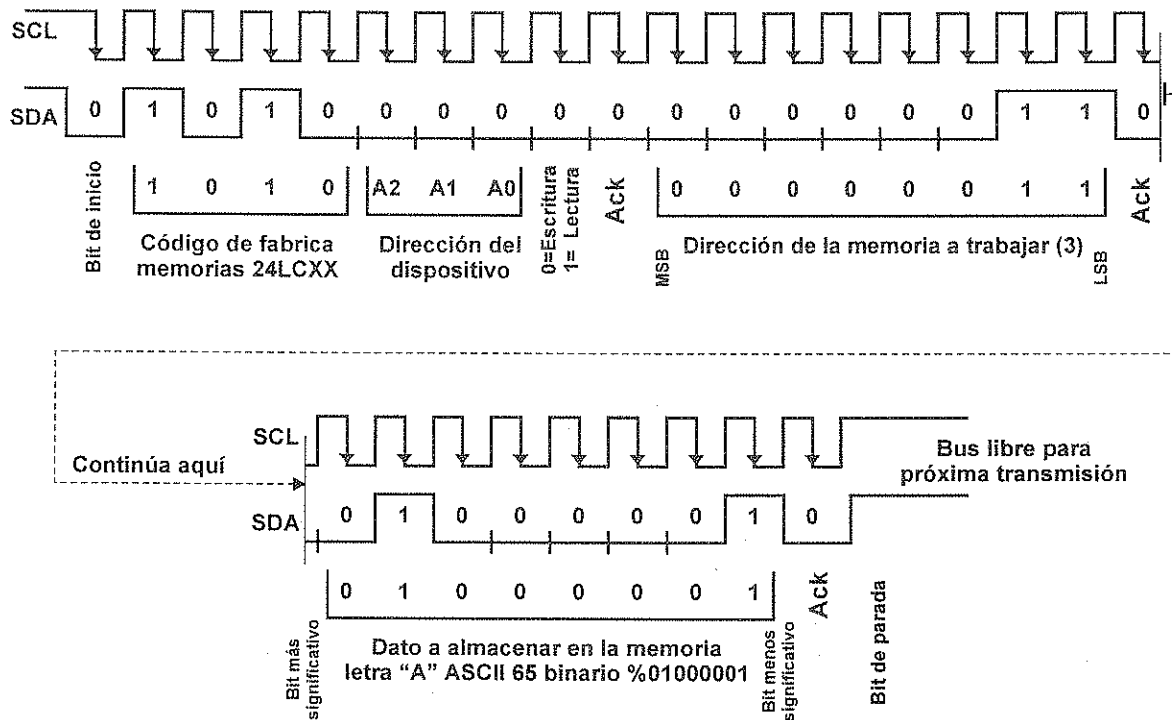


Figura 5.9.12.2. Esquema de una transmisión completa con la interfaz I²C para guardar el número 65 en la dirección 3 de una memoria serial 24LC04B.

Noten que la dirección del CI. A2, A1, A0 es 000, lo que quiere decir que estos tres pines van conectados a tierra, según la tabla 5,9,12,1, se pueden instalar 4 dispositivos de memoria en un bus, estos son empezando por A2, A1 y A0 los siguientes: 00x, 01x, 10x y 11x, por ejemplo: 01x esta memoria debe tener conectado a 5 voltios el pin A1 y su control sería 10100100.

LA DECLARACIÓN I2CWRITE Y I2CREAD. Estas declaraciones sirven para escribir y leer datos en un chip EEPROM serial usando una interfaz I²C de 2 hilos, funcionan en modo I²C Master y también puede ser utilizado para comunicarse con otros dispositivos de interfaz I²C como sensores de temperatura, reloj calendarios, conversores A/D, etc.

Los 7 bits de control contienen el código de fábrica del chip y la selección del chip A2, A1, A0, el último bit es una bandera interna que indica si es un comando de lectura o escritura y **no se debe usar**. Por lo tanto el control para nuestro caso en lectura o escritura es %10100000.

Debido a que los pines SDA y SCL de la memoria 24LC04B son de colector abierto, estas deben ir conectadas con resistencias de 4,7 K Ω pull-up, sin embargo existe una línea de comando que hace que no se necesite la resistencia pull-up del SCL, esta se debe agregar al comienzo del programa:

```
DEFINE I2C_SCLOUT 1 ; no es necesario resistencia pull-up en SCL (reloj)
```

También cabe indicar que existen algunas memorias que necesitan de un período de tiempo para poder ser grabadas, por lo que se adiciona un **PAUSE 10** después de cada grabación.

Su estructura es de la siguiente manera:

```
I2CWRITE portb.6, portb.7, %10100000, 0, [65] ; almacenar en la dirección 0 el dato 65
PAUSE 10 ; pausa necesaria para completar la grabación
```

I2CWRITE pin dato, pin reloj, control, posición memoria, variable.

MATERIALES.

- 1 LCD 2 x 16
- 1 resistencia de 4,7 K Ω
- 1 resistencia de 10 Ω
- 1 potenciómetro de 10 K Ω
- 1 memoria serial 24LC04B de microchip o equivalente
- 1 switch selector de 3 pines.

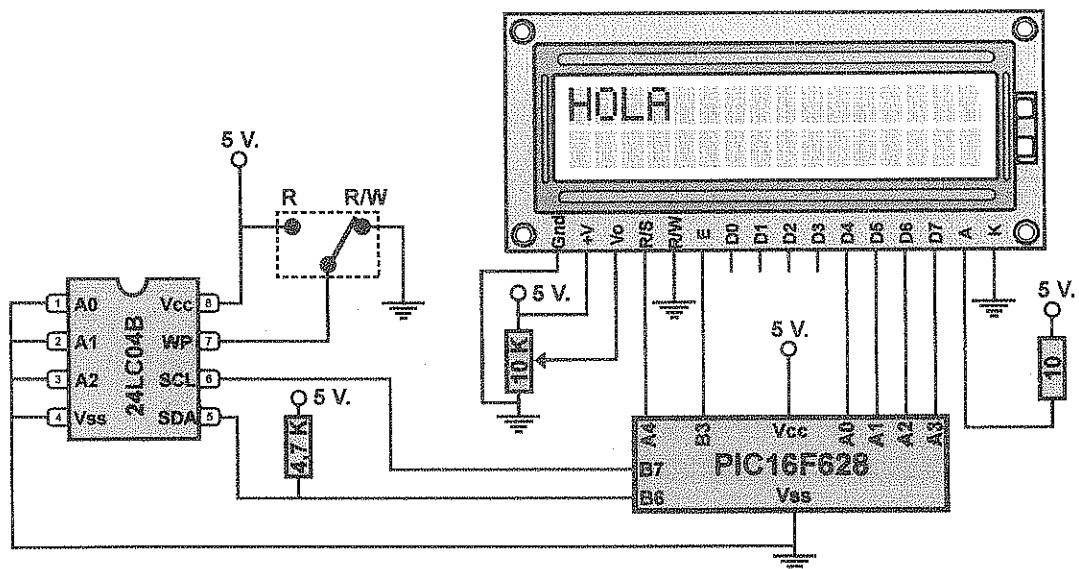


Figura 5.9.12.3. Esquema de conexionado de una memoria 24LCXX a un PIC, el switch selector permite proteger a la memoria de escrituras accidentales al colocar en R Read.

En nuestro caso vamos a escribir y leer en la memoria, por lo que el pin WP debe estar colocado en estado bajo, una vez que se haya grabado se podrá colocar este pin en estado alto para proteger la memoria de futuras escrituras, los datos almacenados en esta memoria permanecen aún si se le corta la alimentación al CI. (no volátil), el acceso a estos datos se lo realiza las veces deseadas, recuerde que la memoria serial soporta 1,000.000 de ciclos de borrado y escritura, por lo tanto se debe tener cuidado de no ejecutar un programa que almacene una y otra vez el mismo dato.

```

DEFINE I2C_SCLOUT 1 ;para que no necesite resistencia pull-up en SCL

contro CON %10100000 ;contro contiene valor constante %10100000
PinSCL VAR Portb.7 ;pin señal de reloj I2C
PinSDA VAR Portb.6 ;pin de datos I2C
dato VAR BYTE ;variable para almacenar dato leído

Inicio:
LCDOUT $fe,1," Grabando..." ;limpiar y sacar el texto en LCD
PAUSE 1000

I2CWRITE PinSDA,PinSCL,contro,0,["H"] ;guarda la H en posición 0
PAUSE 10 ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,1,["O"] ;guarda la O en posición 1
PAUSE 10 ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,2,["L"] ;guarda la L en posición 2
PAUSE 10 ;pause para la grabación
I2CWRITE PinSDA,PinSCL,contro,3,["A"] ;guarda la A en posición 3
PAUSE 10 ;pause para la grabación

LCDOUT $fe,1,"Leer memoria" ;limpiar y sacar el texto en LCD

PAUSE 1000
LCDOUT $FE,1 ;limpiar pantalla del LCD

leer: ;programa para leer la memoria serial.

I2CREAD PinSDA,PinSCL,contro,0,[dato] ;leer la mem. 0 y guardar en dato
LCDOUT ,dato ;mostrar dato en el LCD
PAUSE 1000 ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,1,[dato] ;leer la mem. 1 y guardar en dato
LCDOUT ,dato ;mostrar dato en el LCD
PAUSE 1000 ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,2,[dato] ;leer la mem. 2 y guardar en dato
LCDOUT ,dato ;mostrar dato en el LCD
PAUSE 1000 ;esperar 1 seg.
I2CREAD PinSDA,PinSCL,contro,3,[dato] ;leer la mem. 3 y guardar en dato
LCDOUT ,dato ;mostrar dato en el LCD

END

```

Figura 5.9.12.4. memoria 24LCXX.bas Programa para escribir y leer datos en un chip EEPROM.

5.9.13. COMUNICACIÓN PC CON EL RELOJ CALENDARIO DS1307.

Como habíamos dicho anteriormente el bus I²C, permite la comunicación con algunos dispositivos como las memorias 24CXX, los procesadores de señal, codificadores de video, sensores de temperatura, RTC (reloj en tiempo real), etc. Haremos un proyecto de lectura y escritura de un RTC (Real Time Clock), este proyecto es muy similar al anterior, pero con la diferencia que el byte de control es %11010000, (propio del fabricante), y su modo de grabación de datos es en sistema hexadecimal.

Una aclaración importante es que por ser un experimento los datos a ser almacenados serán: jueves 30 de septiembre del 2004 y la hora 13:32:00, pudiendo ser cambiado a gusto del lector. Estos datos serán grabados una sola vez, ya que se utilizará una condición de bandera, esta será almacenada en la memoria EEPROM del mismo PIC y se llamará (actualizado=1), significa que ya corrió una vez el programa, por consiguiente si se lo vuelve a prender el PIC, no guardará nuevamente los datos, esto tiene la finalidad de que los datos fecha y hora sean una sola vez igualados, para que cuando usted vuelva a encender el PIC, muestre la hora actual.

MATERIALES.

- 1 LCD 2 x 16
- 1 resistencia de 10 Ω
- 1 potenciómetro de 10 K Ω
- 1 RTC DS1307
- 3 resistencias de 4,7 K Ω
- 1 resistencia de 470 Ω
- 1 transistor 2N3904
- 1 led rojo
- 1 oscilador cristal de 32768 HZ (comunes encontrar en los relojes de mano y pared)
- 1 pila de 3 V (CR2032), con su porta pilas ver figura 5.9.13.3.

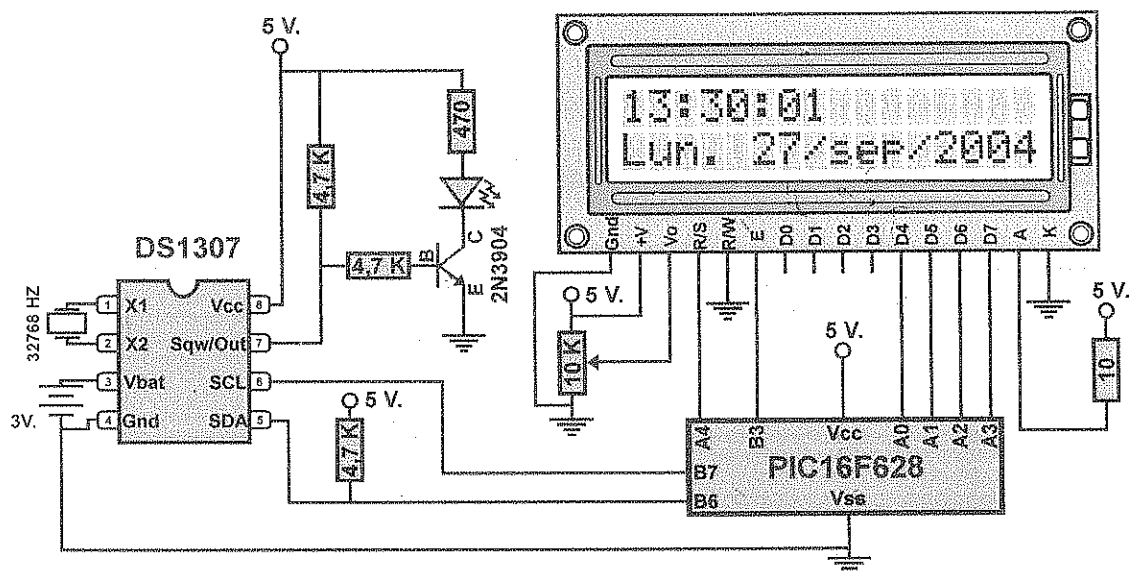


Figura 5.9.13.1. Esquema de conexión de un RTC con el PIC para un reloj calendario.

```

DEFINE I2C_SCLOUT 1 ;para que no necesite resistencia pull-up en SCL

CPIN VAR Portb.7 ;pin señal de reloj I2C
DPIN VAR Portb.6 ;pin de datos I2C

segu VAR BYTE ;definir tamaño de variable segundos 1 a 255
minu VAR BYTE ;variable para los minutos
hora VAR BYTE ;variable para las horas
diaS VAR BYTE ;variable día de la semana
diaF VAR BYTE ;variable día fecha del mes
mes VAR BYTE ;variable mes
anio VAR BYTE ;variable año de 2 dígitos

dato VAR BYTE ;variable para almacenar dato leído
actualizado VAR BIT ;variable para almacenar un 1 o 0

EEPROM 0,[0] ;memoria 0 con el valor inicial 0 , sirve para
;indicar que nunca ha corrido este programa
READ 0,actualizado ;carga el valor de la memoria EEPROM dirección 0

IF actualizado =0 THEN grabarRTC ;si es la 1ra vez que corre ir a grabar RTC
;caso contrario solo leer el RTC

INICIO:
I2CREAD DPIN,CPIN,%11010000,0,[segu] ;leer los datos de mem. 0,
I2CREAD DPIN,CPIN,%11010000,1,[minu] ;1,2,..y guardarlos en sus
I2CREAD DPIN,CPIN,%11010000,2,[hora] ;respectivas variables
I2CREAD DPIN,CPIN,%11010000,3,[diaS]
I2CREAD DPIN,CPIN,%11010000,4,[diaF]
I2CREAD DPIN,CPIN,%11010000,5,[mes]
I2CREAD DPIN,CPIN,%11010000,6,[anio]

LCDOUT $fe,1,HEX2 hora,":", HEX 2 minu,":", HEX 2 segu ;mostrar la hora
; min y segs. en 2 dígitos (HEX2)
LCDOUT $fe,$c0 ;saltar a la 2da línea del LCD
IF diaS=$1 THEN LCDOUT "Dom." ;mostrar día de la semana
IF diaS=$2 THEN LCDOUT "Lun."
IF diaS=$3 THEN LCDOUT "Mar."
IF diaS=$4 THEN LCDOUT "Mie."
IF diaS=$5 THEN LCDOUT "Jue."
IF diaS=$6 THEN LCDOUT "Vie."
IF diaS=$7 THEN LCDOUT "Sab."

LCDOUT $fe,$c5, HEX 2 diaF, "/" ;mostrar el día del mes /
LCDOUT $fe,$cB, "/20", HEX 2 anio ;mostrar año /20 + 04

LCDOUT $fe,$c8 ;pasar a la casilla 8
IF mes=$1 THEN LCDOUT "ene" ;mostrar el mes
IF mes=$2 THEN LCDOUT "feb"
IF mes=$3 THEN LCDOUT "mar"

```

continúa ...


```

IF mes=$4 THEN LCDOUT "abr"
IF mes=$5 THEN LCDOUT "may"
IF mes=$6 THEN LCDOUT "jun"
IF mes=$7 THEN LCDOUT "jul"
IF mes=$8 THEN LCDOUT "ago"
IF mes=$9 THEN LCDOUT "sep"
IF mes=$10 THEN LCDOUT "oct"
IF mes=$11 THEN LCDOUT "nov"
IF mes=$12 THEN LCDOUT "dic"
PAUSE 500                                ;esperar 0,5 segundos

GOTO inicio                              ;volver a leer los datos

;***** subrutina grabar *****
grabarRTC:

I2CWRITE DPIN,CPIN,%11010000,0,[$00]    ;setear 00 segundos
PAUSE 10                                ;retardo para finalizar grabación
I2CWRITE DPIN,CPIN,%11010000,1,[$30]    ;setear 30 minutos
PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,2,[$13]    ;setear las 13 horas
PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,3,[$2]     ;setear día lunes ,D=1,L=2
PAUSE 10                                ;M=3 , M=4, J=5, V=6, S=7
I2CWRITE DPIN,CPIN,%11010000,4,[$27]    ;setear día 27 del mes
PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,5,[$9]     ;setear mes septiembre
PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,6,[$04]    ;setear año 04
PAUSE 10
I2CWRITE DPIN,CPIN,%11010000,7,[$10]    ;control %00010000 para
PAUSE 10                                ;encender el led cada 1 seg.

WRITE 0,1                                ;escribe en la memoria 0 el valor de 1 para que no
;se vuelva a grabar otra vez estos datos en el RTC
GOTO inicio                              ;ir a presentar los datos en el LCD
END

```

Figura 5.9.13.2. reloj DS1307.bas Programa para escribir y leer datos en un RTC DS1307.

Como podrán observar la batería es la que le mantiene en funcionamiento al RTC cuando no hay alimentación DC, por tal razón cuando apagamos todo el circuito, y luego lo volvemos a prender, notamos que el reloj no se ha desigualado, pero si retiramos la batería, el reloj se detiene cuando lo cortamos la alimentación del circuito, y cuando se lo vuelve a conectar, sigue corriendo el tiempo pero continúa en el segundo que se quedó en el instante que se le cortó la alimentación.

El transistor sirve para encender el led con la fuente que alimenta el circuito y se apaga cuando deja de alimentarse el circuito, por lo que la batería solo alimenta al CI. DS1307 mientras no hay alimentación en el pin VCC.

Se debe entender que el PIC está leyendo los datos del RTC cada 0,5 segundos, cuando en realidad debería leer cada segundo. Para mejorar este programa podemos utilizar una interrupción por cambio de estado en el pin B.0, aquí conectamos la señal SQW del RTC para que el PIC ejecute una subrutina de interrupción y lea los datos del RTC exactamente cuando el RTC le diga que transcurrió un segundo, con esto tenemos al PIC disponible para otras aplicaciones y no se quedaría esclavizado al RTC leyendo datos 2 veces por segundo. Para aplicar la interrupción ver literal 5.10.1 Utilizando la interrupción del puerto B.0.

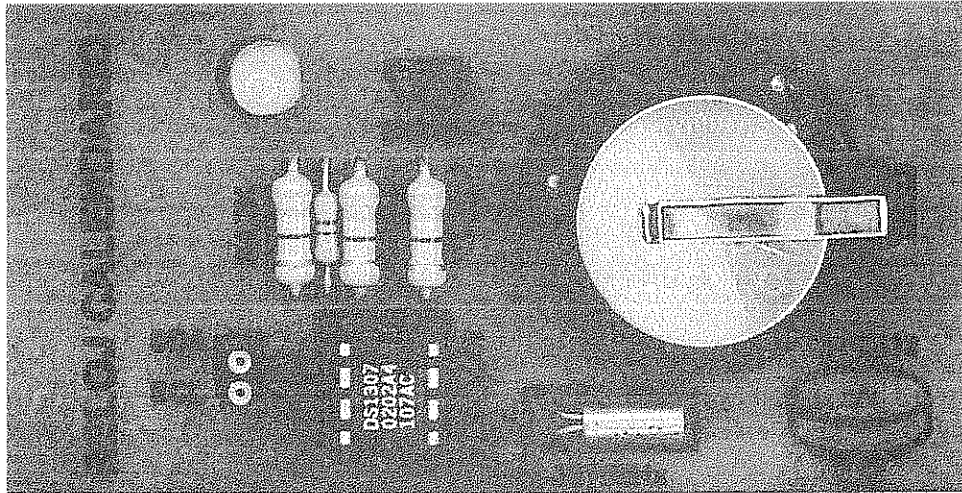


Figura 5.9.13.3. Fotografía de un módulo RTC del entrenador experto de PIC'S EE-01.

5.10 INTERRUPCIONES

5.10.1. UTILIZANDO LA INTERRUPCIÓN DEL PUERTO B.0

Existen aplicaciones en donde un evento es muy importante atenderlo, por ejemplo cuando algún dispositivo intenta comunicarse con el PIC, en un sistema de seguridad en donde una zona es más importante que las demás zonas o como el ejercicio anterior en donde cada segundo envía un pulso SQW, podríamos utilizar este pulso para indicar al PIC que debe leer los nuevos datos (segundo, minutos, hora, etc.).

Como práctica para poder entender la interrupción en el cambio de estado del Portb.0, haremos un parpadeo de un led rojo cada 200ms, y cuando exista una interrupción externa (pulsador), deja de ejecutarse el programa y atiende un Handler (subrutina) el cual contiene un programa en donde se enciende un led verde por 1 segundo, una vez terminado el programa de interrupción, retorna al programa principal en el lugar mismo donde ocurrió la interrupción.

NOTA: También existen otras fuentes de interrupción a más del puerto B.0, como el cambio de estado del puerto B.4 al Puerto B.7, los TIMER0, 1 y 2, por lo que se recomienda leer las hojas de datos del PIC16F628.

LAS DECLARACIONES ON INTERRUPT, DISABLE, RESUME Y ENABLE. Todas estas declaraciones sirven para ejecutar un handler (subrutina con RESUME) de interrupción.

ON INTERRUPT GOTO prog2; quiere decir en caso de darse una interrupción suspender el programa actual e ir a prog2.

DISABLE; sirve para deshabilitar la interrupción, en caso de que no deseemos que atienda la interrupción, como por ejemplo cuando ya está atendiendo una interrupción.

RESUME; equivale al RETURN de un GOSUB, en donde retorna a la línea del programa en donde ocurrió la interrupción.

ENABLE; quiere decir habilitar nuevamente la interrupción, después de esto todas las interrupciones, son atendidas.

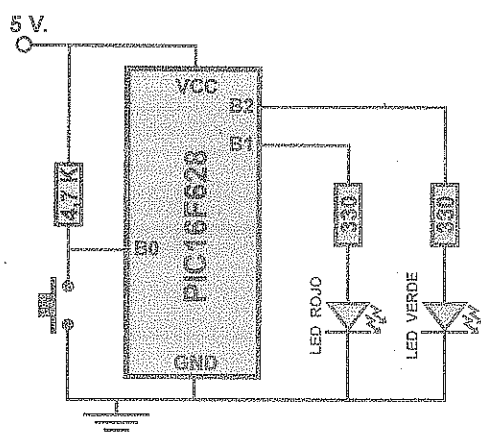


Figura 5.10.1.1. Conexión de 2 leds y un pulsador para la práctica de interrupción.

MATERIALES.

- 1 led rojo 5 mm.
- 1 led verde 5mm.
- 2 resistencias de 330 Ω
- 1 resistencia de 4,7 K Ω
- 1 pulsador NA.

```
                                ;programa para manejar una interrupción en el p B.0
LED VAR PORTB.1
LED2 VAR PORTB.2

ON INTERRUPT GOTO verde        ; en caso de existir una interrupción ir a verde
INTCON = %10010000            ;habilita la interrupción B.0

PROG:                          ;programa principal
    HIGH LED                   ;encender el led rojo
    PAUSE 200
    LOW LED
    PAUSE 200
    GOTO PROG                  ;mantener en este lazo

    DISABLE                    ;deshabilita las interrupciones en el handler
verde:                          ;handler de la interrupción
    HIGH LED2
    PAUSE 2000
    LOW LED2
    INTCON = %10010000        ;habilita la interrupción B.0
    RESUME                    ;retorna a programa principal donde ocurrió la inte.
    ENABLE                    ;Habilita las interrupciones después del handler
END
```

Figura 5.10.1.2. interrupt.bas Programa para practicar con la interrupción en el puerto B.0.

Como pudieron observar, en el momento que se presiona el pulsador, automáticamente sale del programa y atiende el handler de interrupción llamado verde, finalizado esto el **RESUME** lo retorna al lugar en donde ocurrió la interrupción, sin embargo se puede poner **RESUME prog3**, esta vez va a una subrutina prog3 e ignora el retorno al lugar de la interrupción, otra recomendación importante es que si queremos atender rápidamente una interrupción, no debemos poner **PAUSES** muy largos como por ejemplo **PAUSE 10000**, equivalente a 10 segundos, si se da la interrupción en la mitad del pause (5 segundos), deberá esperar a que termine el pause para ir al handler de interrupción, es decir los 5 segundos restantes, lo mejor para estos casos es encerrar el pause en lazos **FOR... NEXT**, de la siguiente manera:

```
FOR x = 1 TO 100                ; repetir 100 veces (equivale a 10 segundos)
PAUSE 100
NEXT
```

Por lo que la atención al handler de interrupción será en 100 mls después de la interrupción.

Cabe también indicar que en esta práctica la interrupción se da solo cuando existe un cambio de estado de 0 a 1 en el puerto B0, por lo que se habrán dado cuenta que si mantienen pulsado la tecla, no se genera la interrupción, sino cuando la soltamos, para que la interrupción se genere en el flanco de bajada, es decir cambio de estado de 1 a 0, debemos adicionar después de `INTCON=%10010000`, la siguiente línea para modificar el bit INTEDG del registro OPTION:

<code>OPTION_REG=0</code> ;modifica el bit 6 del registro OPTION, activa en flanco de bajada a B.0
--

5.10.2. UTILIZANDO LA INTERRUPCIÓN DEL PUERTO B.4 AL B.7.

Para trabajar con la interrupción por cambio de estado del puerto b.4 al puerto b.7, podemos utilizar el mismo programa 5.10.1.2, solamente cambiando las 2 líneas que dicen `INTCON=%10010000` por `INTCON=%10001000`, que quiere decir habilitar la interrupción de los puertos B.4 al B.7, para mayor información revise en los Data sheets el registro INTCON.

Para el diagrama de conexión debemos colocar 4 pulsadores en los puertos B.4 al B.7, no se debe dejar sueltos estos pines ya que produce errores y salta al handler de interrupción en cualquier momento. Se puede desconectar el pulsador del puerto B.0, ya que no se está atendiendo esta interrupción.

5.10.3. RELOJ DIGITAL UTILIZANDO LA INTERRUPCIÓN DEL TMR0.

El TMR0, es una valiosa herramienta que disponen los PIC'S, para el caso del PIC16F628, dispone de 3 TIMERS, 1 de 16 bits (TMR1) y 2 de 8 bits, los TMR0 y TMR2, la calibración para estos contadores, se dan en el registro OPTION, aquí se puede seleccionar si el incremento es con flanco de subida o de bajada y si la fuente es externa (pin A4/TOCKI) o interna (Oscilador), en nuestro caso será interna generada por el Oscilador. Cuando el conteo del temporizador TMR0, llega a 256 y pasa a 0, se genera una interrupción, para que esto suceda se debe habilitar el registro INTCON bit 7 (GIE = 1), y también el bit 5 (TOIE =1), quedando así:

<code>INTCON=%10100000</code> ; habilita interrupción por TMR0
--

En el registro OPTION se debe definir la rata del preescalador, se debe poner:

<code>OPTION_REG=%01010110</code> ; preescalador 1:128, asignado al TMR0 y ciclo de reloj interno.
--

Para poder entender mejor estos registros y sus funciones tenga a la mano la hoja de datos.

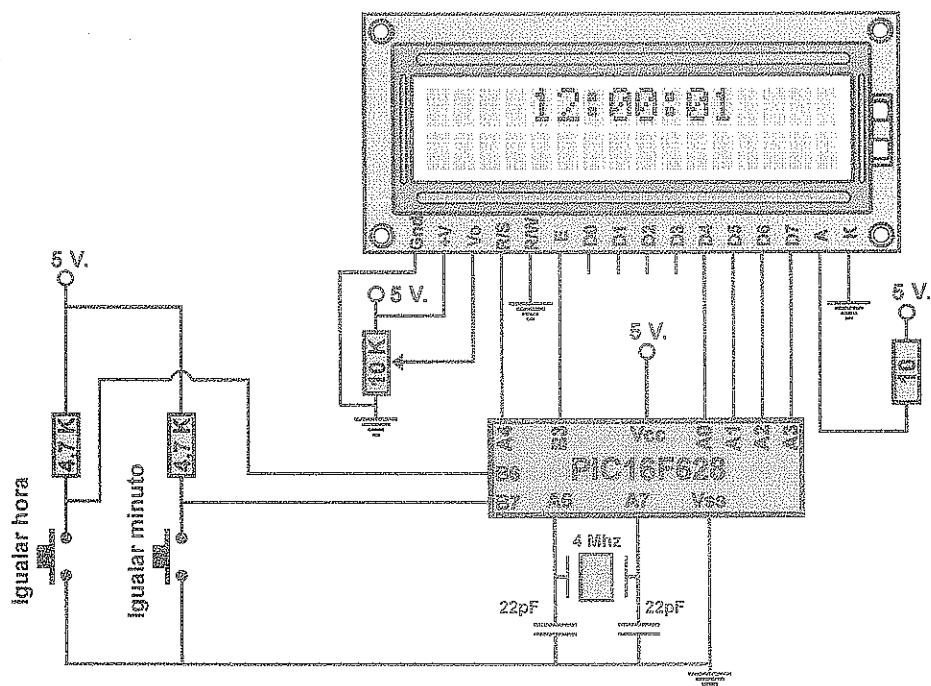
La práctica a realizarse consiste en ejecutar una interrupción, cada que el contador del TMR0, llega a 256, pero no empieza desde 0 ya que se le asignó un valor inicial de 4 (ver la línea # 70 del programa TMR0-reloj.bas), por lo que el tiempo sería $128 \times 252 = 32256$ Us y esto repetido 31 veces, conseguimos acercarnos más al tiempo de 1 segundo ($32256 \text{ Us} \times 31 = 999936 \text{ Us}$), luego de esto se incrementa la variable `segun = segun +1` y se actualiza el LCD.

El proyecto dispone de 2 pulsadores para poder igualar la hora, uno aumenta los minutos y otro aumenta las horas, la gran desventaja de este proyecto es que si se corta la alimentación

del circuito, se resetea la hora (12:00:00), lo que no sucede con un RTC.

MATERIALES.

- 1 LCD 2 x 16
- 1 resistencia de $10\ \Omega$
- 1 potenciómetro de $10\ K\ \Omega$
- 2 resistencias de $4,7\ K\ \Omega$
- 2 pulsadores NA
- 1 oscilador cristal de 4 MHZ
- 2 capacitores cerámicos de 22pF.




```

hora = 12                                ;setea tiempo a 12:00:00
minut = 0
segun = 0
cuenta = 0
actual = 1

OPTION_REG = %1010110                    ;setea TMR0 de interrupción cada 32768 microsegundos
INTCON = %10100000                        ;setea TMR0 configurando y habilita PORTB pullups
ON INTERRUPT GOTO interrup                ;habilita TMR0 interrupción interna

INICIO:
    IF PORTB.7 = 0 THEN incmin             ;botón para igualar minutos
    IF PORTB.6 = 0 THEN inchr              ;botón para igualar horas

actualiza: IF actual = 1 THEN              ;chequea si hay que actualizar LCD

    LCDOUT $fe,1," ",dec2 hora,".",dec2 minut,".",DEC2 segun

    actual = 0                             ;pantalla actualizada
    ENDIF
    GOTO INICIO

; ***** para igualar la hora *****
incmin: minut = minut + 1
    IF minut >= 60 THEN minut=0
    GOTO pausa

inchr: hora = hora + 1
    IF hora >= 24 THEN hora=0
    GOTO pausa

pausa: FOR x = 1 TO 20                     ;retardo de 200 mls
    PAUSE 10                               ;pasos de 10 mls para no perder interrupciones
    NEXT x
    actual = 1                             ;indica actualizar pantalla LCD
    GOTO actualiza

; ***** Handler de interrupciones para incrementar contador *****
DISABLE                                   ;deshabilita interrupciones durante el proceso
interrup:
    cuenta = cuenta + 1                    ;cuenta las interrupciones del TMR0
    TMR0=4                                 ;resta 4 al contador de 256 del TMR0

    IF cuenta < 31 THEN reset               ;31 cuentas (32256ms = 999936uS)
    cuenta = 0
    segun = segun + 1
    IF segun >= 60 THEN
        segun = 0
        minut = minut + 1

```

continúa ...

```

    IF minut >= 60 THEN
        minut = 0
        hora = hora + 1
    IF hora >= 24 THEN
        hora = 0
    ENDIF
    ENDIF
    ENDIF
actual = 1                                ;actualizar LCD

reset:
    INTCON.2 = 0                          ;resetea la bandera de interrupción del TMR0

RESUME
END

```

Figura 5.10.3.2. TMR0-reloj.bas Programa para practicar con la interrupción del TMR0.

5.11 CONVERTOR A/D

5.11.1. CONVERTOR ANALÓGICO DIGITAL DEL PIC 16F81X.

Esta nueva familia de PIC'S los 16F818 y 16F819, iniciaron su aparición a mediados del 2004, por lo que será muy común encontrarlos en las tiendas electrónicas ya que posee 5 convertidores A/D de 10 bits c/u. y su memoria RAM y EEPROM, son más extensas que la del PIC16F628, y además su costo es más inferior que los PIC16F62X y PIC16F84A.

	PIC16F84A	PIC16F628	PIC16F819
Memoria de programa	1024	2048	2048
Memoria datos EEPROM	64	128	256
Memoria RAM	68	224	256
Pines de entrada/salida	13	16	16
Comparadores	-	2	-
Convertidores A/D	-	-	5

Figura 5.11.1.1. Tabla de comparación entre el PIC16F84A, el PIC16F628 y el PIC16F819.

El PIC16F81X al igual que el PIC16F62X, disponen de oscilador interno, pero el PIC16F81X dispone de 8 calibraciones para el oscilador interno, desde 31,25 KHZ hasta 8 MHZ.

Para poder empezar a practicar con este nuevo PIC, haremos un ejercicio de lectura de un convertidor A/D del PIC16F819, su funcionamiento es muy simple, un pin del PIC, los del puerto A (A0, A1, A2, A3 y A4), son capaces de detectar el nivel de voltaje que ingresan a un pin, por ejemplo: si utilizamos un convertidor A/D de 8 bits, quiere decir que entre los voltajes de referencias, digamos que $V_{ref-}=0V$. y $V_{ref+}=5V$., los 5 V. los dividirá en 255 segmentos, (19,6 mV.), entonces si a la entrada del pin A/D ingresan 19,6 mV, el registro ADCIN nos entregará un valor de 1, y así tenemos que para:

0V ADCIN = 0
2,49V ADCIN = 127
5V ADCIN = 255

En definitiva nos dice que voltaje está entrando por un pin pero en valores de una variable de 255, para poder entender mejor haremos la siguiente práctica que consiste en calibrar el oscilador interno a 4 MHZ, leer el canal 0 a 8 bit (puede leer a 10 bits) el voltaje que ingresa y mostrarlo en un LCD.

Para poner en funcionamiento los convertidores A/D, debemos cambiar unos bits del registro ADCON1, ver tabla de la figura 5.11.1.2., en donde se debe configurar cual es el canal o los canales que deseamos utilizar, así como también cuales son los voltajes de referencia. En nuestro caso utilizaremos $ADCON1 = \%00001110$, que significa que solo vamos a utilizar el canal 0 (A.0) con voltajes de referencias positivo y negativos, los mismos que utiliza el PIC para su alimentación, es decir en este caso 0 y 5V.

MATERIALES.

- 1 PIC16F819
- 1 LCD 2 x 16
- 1 resistencia de 10 Ω
- 2 potenciómetros de 10 K Ω .

PCFG	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C/R
0000	A	A	A	A	A	AVDD	AVSS	5/0
0001	A	VREF+	A	A	A	AN3	AVSS	4/1
0010	A	A	A	A	A	AVDD	AVSS	5/0
0011	A	VREF+	A	A	A	AN3	AVSS	4/1
0100	D	A	D	A	A	AVDD	AVSS	3/0
0101	D	VREF+	D	A	A	AN3	AVSS	2/1
011x	D	D	D	D	D	AVDD	AVSS	0/0
1000	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1001	A	A	A	A	A	AVDD	AVSS	5/0
1010	A	VREF+	A	A	A	AN3	AVSS	4/1
1011	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1100	A	VREF+	VREF-	A	A	AN3	AN2	3/2
1101	D	VREF+	VREF-	A	A	AN3	AN2	2/2
1110	D	D	D	D	A	AVDD	AVSS	1/0
1111	D	VREF+	VREF-	D	A	AN3	AN2	1/2

D=Digital A= Análogo

Figura 5.11.1.2. Tabla de configuración para el registro ADCON1, noten que ADCON1=7 corresponde a PCFG =011x, por lo tanto todo el puerto A se convierte en pines digitales, algo parecido ocurría con el CMCON=7 del PIC16F628, este apagaba los comparadores de voltaje y los volvía en pines digitales.

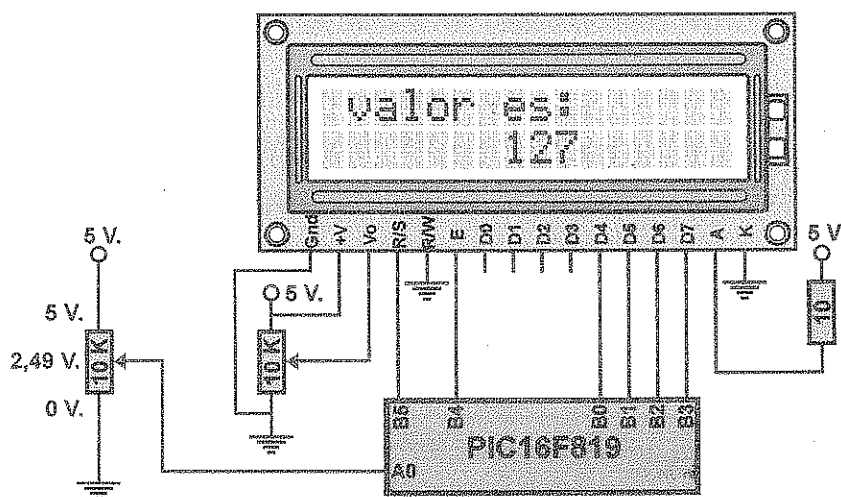


Figura 5.11.1.3. Conexión de un potenciómetro de 10K para obtener un divisor de voltaje desde 0V. hasta 5 V., trate de poner el potenciómetro en 2,49 V. y observará en el LCD el dato 127.

Otro cambio importante es que los pines que controlan al LCD, no pueden ser los predefinidos por pbb, por lo tanto debemos cambiarlos de posición utilizando **DEFINE**.

```

;calibraciones  %111 8MHZ  %110 4MHZ  %101 2MHZ  %100 1MHZ  %011 500KHZ
;%010 250KHZ  %001 125KHZ  %000 31.25KHZ adicionarle a todos 0100

OSCCON=%1100100                                ; calibra oscilador interno a 4 MHz

DEFINE      LCD_DREG  PORTB                      ;bit de datos del LCD empezando
DEFINE      LCD_DBIT  0                          ;por B.0 ,B.1, B.2 y B.3
DEFINE      LCD_RSREG PORTB                      ;bit de registro del LCD conectar
DEFINE      LCD_RSBIT 5                          ;en el puerto B.5
DEFINE      LCD_EREG  PORTB                      ;bit de Enable conectar en el
DEFINE      LCD_EBIT  4                          ;puerto B.4

DEFINE      ADC_BITS  8                          ;Fija número de bits del resultado (5,8,10)
DEFINE      ADC_CLOCK 3                          ;Fije EL CLOCK (rc = 3)
DEFINE      ADC_SAMPLEUS 50                      ;Fije el tiempo de muestreo en Us
;ADC_SAMPLEUS es el número de microsegundos que el programa espera
;entre fijar el canal y comenzar la conversión analoga/digital.

TRISA=%1                                          ;el puerto A.0 es de entrada
ADCON1=%00001110                               ;el puerto A.0 es conversor los demás Digitales

datos VAR BYTE                                  ;crear variable datos para guardar el resultado
PAUSE 500                                        ;esperar 0.5 seg.

inicio:
    ADCIN 0, datos                              ;leer el canal 0 y guardarlo en datos
    LCDOUT $fe, 1, " valor es:"
    LCDOUT $fe,$c7,DEC datos                    ;desplegar el valor de datos en decimal

    PAUSE 300

    GOTO inicio                                ;volver a medir el conversor A/D
END

```

Figura 5.11.1.4. AD-16F819-8.bas Programa para practicar con el conversor analógico digital.

5.11.2. CONVERSOR ANALÓGICO DIGITAL DEL PIC 16F87X.

En esta práctica utilizaremos 3 conversores A/D de los 8 que dispone el PIC16F877A, este PIC se caracteriza por tener 40 pines, de los cuales 33 son puertos de entrada/salida, una memoria FLASH de 8192 palabras, una RAM de 368 bytes y una EEPROM de 256 bytes, por lo que este PIC está destinado para proyectos grandes. Esta práctica nos servirá para poder familiarizarnos con PIC'S de mayor capacidad, cabe recalcar que este PIC necesita oscilador externo, ya que no posee oscilador interno, también debemos utilizar en el MCLR una resistencia Pull-Up, ya que no hay forma de deshabilitarlo como sucede con el PIC16F62X.

PCFG3: PCFG0	AN7 ⁽¹⁾ RE2	AN6 ⁽¹⁾ RE1	AN5 ⁽¹⁾ RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs ⁽²⁾
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

D=Digital A= Análogo

Figura 5.11.2.1. Tabla de configuración para el registro ADCON1 del PIC16F877A, noten que ADCON1=7 convierte en pines digitales todos los pines del puerto A y puerto E.

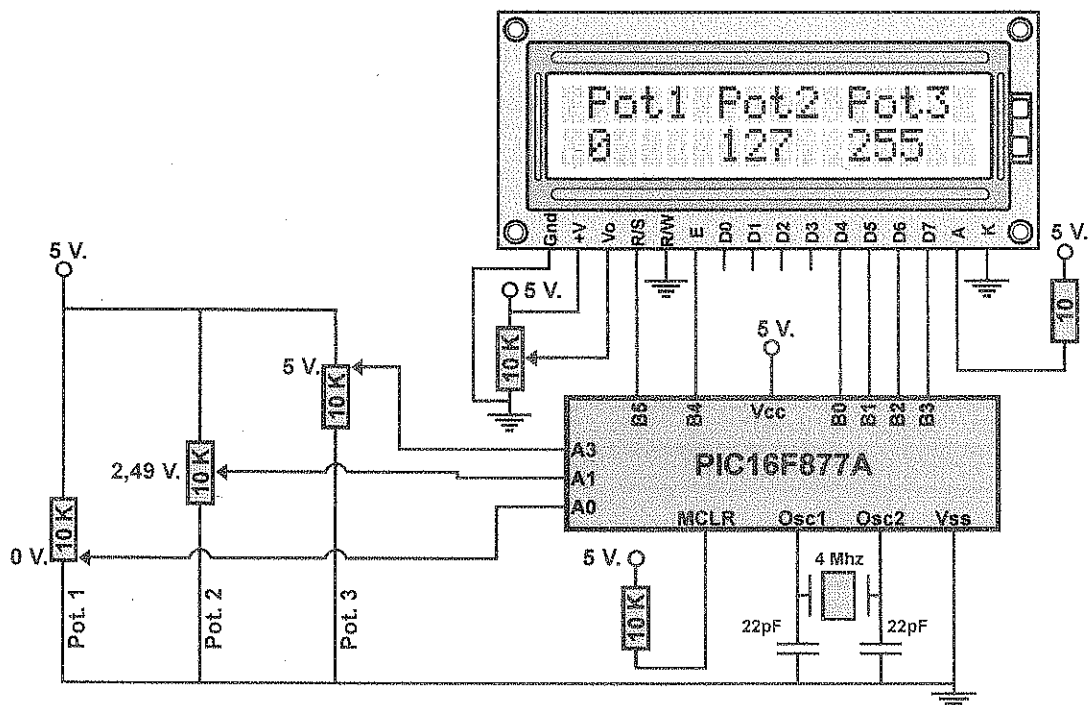


Figura 5.11.2.2. Esquema de conexión de un PIC16F877A para medir los 3 voltajes que ingresan por el divisor de voltaje de cada potenciómetro.

MATERIALES.

- 1 PIC16F877A
- 1 LCD 2 x 16
- 1 resistencia de 10 Ω
- 1 resistencia de 10 K Ω
- 4 potenciómetros de 10 K Ω
- 1 oscilador cristal de 4 MHZ
- 2 condensadores cerámicos de 22pF.

```
DEFINE LCD_DREG PORTB ;bit de datos del LCD empezando
DEFINE LCD_DBIT 0 ;por B.0 ,B.1, B.2 y B.3
DEFINE LCD_RSREG PORTB ;bit de registro del LCD conectar
DEFINE LCD_RSBIT 5 ;en el puerto B.5
DEFINE LCD_EREG PORTB ;bit de Enable conectar en el
DEFINE LCD_EBIT 4 ;puerto B.4
p1 VAR BYTE ;variable para almacenar potenciómetro 1
p2 VAR BYTE ;variable para almacenar potenciómetro 2
p3 VAR BYTE ;variable para almacenar potenciómetro 3
ADCON1 =%100 ;configura PortA 0,1, 3 en conversores A/D

Inicio:
    PAUSE 300
poten1:
    ADCON0 =%1000001 ;activar canal 0 a Fosc/8
    GOSUB medir
    p1= ADRESH
poten2:
    ADCON0 =%1001001 ;activar canal 1 a Fosc/8
    GOSUB medir
    p2= ADRESH
poten3:
    ADCON0 =%1011001 ;activar canal 3 a Fosc/8
    GOSUB medir
    p3= ADRESH

LCDOUT $fe, 1, " Pot1 Pot2 Pot3"
LCDOUT $fe,$a1,#p1 ;casilla 1 el valor decimal de p1
LCDOUT $fe,$c6,#p2 ;casilla 7 el valor decimal de p2
LCDOUT $fe,$cb,#p3 ;casilla 12 el valor decimal de p3
GOTO inicio

medir:
    PAUSEUS 50 ;subrutina para leer el conversor A/D
    ADCON0.2 = 1 ;pausa para seleccionar el canal
    ;iniciar conversión
    PAUSEUS 50 ;pausa para la conversión
    RETURN ;retornar al GOSUB que lo envió
END
```

Figura 5.11.2.3. AD-3-16F877A.bas Programa para practicar con 3 conversores A/D.

5.11.3. TERMÓMETRO DIGITAL CON EL PIC 16F877A.

Para esta práctica necesitaremos el sensor de temperatura LM35, este dispositivo presenta en su pin OUT una variación de 10 mV por grado centígrado, su alimentación puede ser de 4 a 30 Voltios, y su rango de temperatura a sensar entre -55°C hasta 150 °C.

Su funcionamiento es muy simple, primero el voltaje del pin out del LM35, lo vamos a duplicar a través de un juego de amplificadores operacionales (LM358), el cual está configurado como amplificador no inversor, la salida del A.O. LM358, lo conectamos al puerto A.0, el cual está configurado como conversor A/D a 8 bits, este dato se almacena en la variable dato, el cual lo dividimos para 2, con la finalidad de estabilizar la señal que ingresa del pin OUT del LM35. Si la temperatura permanece entre 22°C y 26°C, ninguno de los rele se activa, si la temperatura no se encuentra entre estos 2 rangos, se activa el relé que le corresponde, sea para calentar o enfriar el ambiente, si deseamos modificar los rangos de temperatura, presionamos el pulsador E, con los otros 2 botones aumentamos o disminuimos la temperatura mínima a comparar, y una vez que estemos de acuerdo presionamos la tecla E nuevamente, luego nos pide programar la temperatura máxima, procedemos igual que el caso anterior y cuando presionemos la tecla E, parpadeará tres veces el led, indicando que los nuevos valores ya fueron guardados en la memoria no volátil.

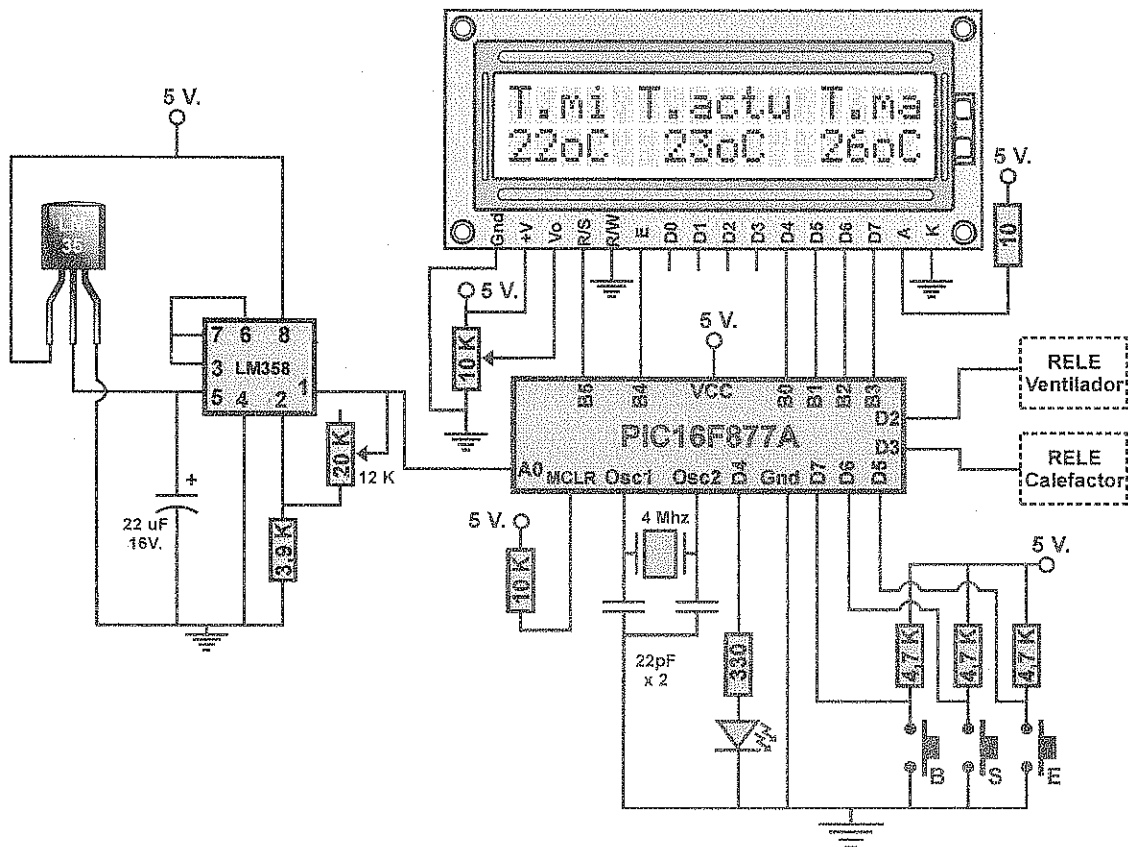


Figura 5.11.3.1. Esquema de conexión de un PIC16F877A para medir la temperatura ambiental.

MATERIALES.

- 1 PIC16F877A
- 1 LCD 2 x 16
- 1 resistencia de 10 Ω

- 1 potenciómetro de 10 K Ω
- 1 potenciómetro de 20 K Ω
- 1 sensor de temperatura LM35
- 1 amplificador O. LM358
- 1 capacitor electrolítico de 22 μ F/16V.
- 1 resistencia de 10 K Ω
- 5 resistencias de 4,7 K Ω
- 1 resistencia de 3,9 K Ω
- 1 resistencia de 330 Ω
- 1 led
- 1 cristal oscilador de 4 MHZ
- 2 condensadores cerámicos de 22pF
- 2 relés 12 V.
- 2 diodos rectificadores 1N4007
- 2 transistores 2N3904.

```

DEFINE LCD_DREG PORTB ;bit de datos del LCD empezando
DEFINE LCD_DBIT 0 ;por B.0 ,B.1, B.2 y B.3
DEFINE LCD_RSREG PORTB ;bit de registro del LCD conectar
DEFINE LCD_RSBIT 5 ;en el puerto B.5
DEFINE LCD_EREG PORTB ;bit de Enable conectar en el
DEFINE LCD_EBIT 4 ;puerto B.4

DEFINE ADC_BITS 8 ;Fije número de BITS del resultado (5,8,10)
DEFINE ADC_CLOCK 3 ;Fije EL CLOCK (rc = 3)
DEFINE ADC_SAMPLEUS 50 ;Fije el tiempo de muestreo en Us
;ADC_SAMPLEUS es el número de microsegundos que el programa espera
;entre fijar el canal y comenzar la conversión analógica/digital.

TRISA =%1 ;el puerto A es de entrada
ADCON1 = %00001110 ;el puerto A.0 es conversor los demás Digitales

dato VAR BYTE ;crear variable dato para guardar
tempbaj VAR BYTE
tempalt VAR BYTE
x VAR BYTE

releF VAR portD.2 ;nombres para los pines
releC VAR portD.3
led VAR portD.4
enter VAR portD.5
bsubir VAR portD.6
bbajar VAR portD.7

EEPROM 0,[22,26] ;contenido inicial de la EEPROM

inicio: ;3 parpadeos del led que indica funciona
FOR x =1 TO 3
HIGH led

```

continúa ...


```

PAUSE 200
LOW led
PAUSE 200
NEXT

READ 0,tempbaj           ;lee la EEPROM 0 y lo guarda en tempbaj
READ 1,tempalt           ;lee la EEPROM 1 y lo guarda en tempalt

sensar:
  ADCIN 0, dato           ;leer el canal 0 (A0) y guarde en dato
  LCDOUT $fe, 1, "T.mi T.actu T.ma" ;limpiar LCD y sacar texto
  dato = dato /2          ;el dato dividir para 2
  LCDOUT $fe,$c6,DEC dato,"oC"      ;Display el decimal de dato
  LCDOUT $fe,$c0,DEC tempbaj,"oC"   ;Display el decimal de tempbaj
  LCDOUT $fe,$cc,DEC tempalt,"oC"   ;Display el decimal de tempalt

  FOR x = 1 TO 50          ;repetir 50 veces
  IF enter =0 THEN grabarla
  PAUSE 10
  NEXT

  IF dato < tempbaj THEN calentar    ;si dato es<tempbaj ir a calentar
  IF dato > tempalt THEN enfriar
  LOW releC : LOW releF             ;apagar los 2 relés
  GOTO sensar                       ;continuar sensando

calentar:
HIGH releC : LOW releF
GOTO sensar

enfriar:
HIGH releF : LOW releC
GOTO sensar

grabarla:
GOSUB soltar

grabarl:
  LCDOUT $fe, 1, "Programar temp."
  LCDOUT $fe,$c0,"baja= ",DEC tempbaj, " oC"
  PAUSE 100
  IF bbajar=0 THEN restarl
  IF bsubir=0 THEN sumar1
  IF enter=0 THEN grabarA
  GOTO grabarl

restarl:
  GOSUB soltar           ;programa antirrebote de tecla
  IF tempbaj < 1 THEN grabarl
  tempbaj= tempbaj -1

```

continúa ...


```

GOTO grabar1
sumar1:
  GOSUB soltar
  IF tempbaj > 40 THEN grabar1
  tempbaj= tempbaj + 1
GOTO grabar1

grabarA:
  GOSUB soltar
  WRITE 0,tempbaj           ;escribir en la dirección 0 de la EEPROM

grabar2:
  LCDOUT $fe, 1 ."Programar temp."
  LCDOUT $fe,$c0,"alta= ",dec tempalt, " oC"
  PAUSE 100
  IF bbajar=0 THEN restar2
  IF bsubir=0 THEN sumar2
  IF enter=0 THEN grabarB
GOTO grabar2

restar2:
  GOSUB soltar
  IF tempalt < 5 THEN grabar2
  tempalt= tempalt -1
GOTO grabar2

sumar2:
  GOSUB soltar
  IF tempalt > 50 THEN grabar2
  tempalt= tempalt + 1
GOTO grabar2

grabarB:
  GOSUB soltar
  WRITE 1,tempalt           ;escribir en la dirección 1 de la EEPROM
GOTO inicio

soltar:
  HIGH led
  PAUSE 150
  LOW led
soltar2:
  IF bbajar=0 THEN soltar2
  IF bsubir=0 THEN soltar2
  IF enter=0 THEN soltar2
  PAUSE 100
  RETURN
END

```

Figura 5.11.3.2. LM35-temp-16F877A.bas Programa para medir la temperatura ambiental.

5.12 UTILIZANDO EL PIC12F6XX

5.12.1. PARPADEO DE LEDS EN EL PUERTO GPIO.1.

En ocasiones existen proyectos en los cuales no se necesitan más de 4 o 5 pines del PIC, un PIC de 16 I/O sería un desperdicio, por tal razón se incluye un pequeño ejercicio de un parpadeo de leds en el puerto gpio, el objetivo de esta práctica es familiarizarnos con esta familia de PIC'S.

	PIC12F629	PIC12F675
Memoria de programa	1024	1024
Memoria datos EEPROM	128	128
Memoria RAM	64	64
Pines de entrada/salida	6	6
Comparadores	1	1
Conversores A/D	-	4

Figura 5.12.1.1. Tabla de comparación entre el PIC16F629 y el PIC12F675.

Como podemos observar la única diferencia entre estas dos subfamilias es que el uno dispone de conversores A/D y el otro no, el ejercicio haremos para el PIC12F675 y para hacer que funcione en un PIC12F629, solo debemos eliminar la línea que configura el convertor A/D ANSEL=0, ya que este último no dispone de tales conversores A/D, su oscilador interno RC da una frecuencia de 4 MHZ, sin embargo podemos utilizar un oscilador externo de hasta 20 MHZ.

MATERIALES.

- 1 PIC12F629 o PIC12F675
- 5 resistencias de 330 Ω
- 5 diodos leds.

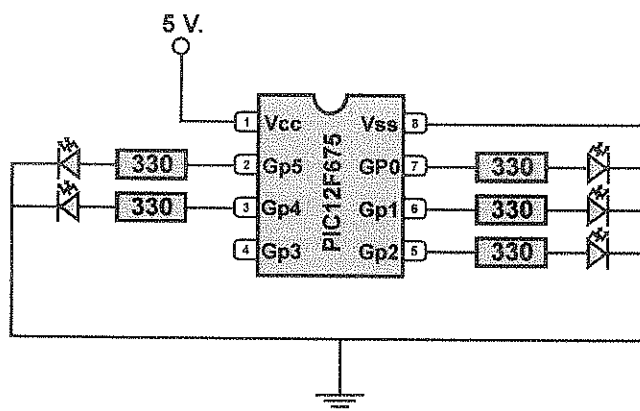


Figura 5.12.1.2. Esquema de conexión del PIC12F6XX, el GPIO.3 es de colector abierto(MCLR)

En este caso el dato leído es 203Ah (anótelo en un papel porque si usted pone borrar el PIC, este dato se puede perder), el programa IC-prog pregunta si quiere utilizar el valor 3FFFh, usted debe poner NO, para que el valor de calibración que le suministra el fabricante sea aceptado, caso contrario si usted presiona SI, está poniendo otro valor de calibración para el oscilador interno.

Si el valor de calibración se le ha borrado accidentalmente, y si usted anotó el valor en este caso 203Ah, puede ayudarlo a colocar en su sitio, escribiéndolo en el programa directamente en la línea:

Dirección-Código programa

```

.....
03E8: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF  yyyyyyyy
03F0: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF  yyyyyyyy
03F8: 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 3FFF 203A  yyyyyyy:

```

CAPÍTULO 6

MÉTODO DE FABRICACIÓN DE CIRCUITOS IMPRESOS

Una vez que usted ha probado su proyecto, pueda que le interese hacer una placa de circuito impreso como las que hemos visto a lo largo del capítulo 5, y talvez un chasis para el mismo, en este capítulo aprenderemos trucos para dar una buena presentación a nuestros proyectos y trabajos, se propone un método revolucionario y muy sencillo comparado con los métodos de dibujo con marcador para circuito impreso, revelado y serigrafía, este es la *transferencia térmica*.

Lo primero que se recomienda es que tengan todos los elementos electrónicos ya comprados y listos, para no tener el inconveniente de que una vez hecho la placa no encuentren el elemento correcto para soldarlo.

Segundo verifique en un protoboard si el proyecto funciona correctamente con todos los elementos que van a soldar en la placa.

Tercero trate de ver los pines que más se le acomoden a las pistas, que no se crucen mucho, que además estén lo más cerca al periférico a manejar.

6.1 DISEÑO DEL CIRCUITO IMPRESO POR SOFTWARE.

Lo primero que necesitamos para hacer nuestra placa es un dibujo de las pistas para nuestros elementos, esto lo conseguimos con la ayuda de un computador, y de los softwares como PROTEL, TANGO, EAGLE, ORCAD, COREL DRAW, o cualquier software de dibujo en el que usted pueda trazar las líneas y pads del circuito (Paint, Photo SHOP, photo Express, etc.), a continuación el circuito de las pistas y el screen de elementos, ya realizados en un software:

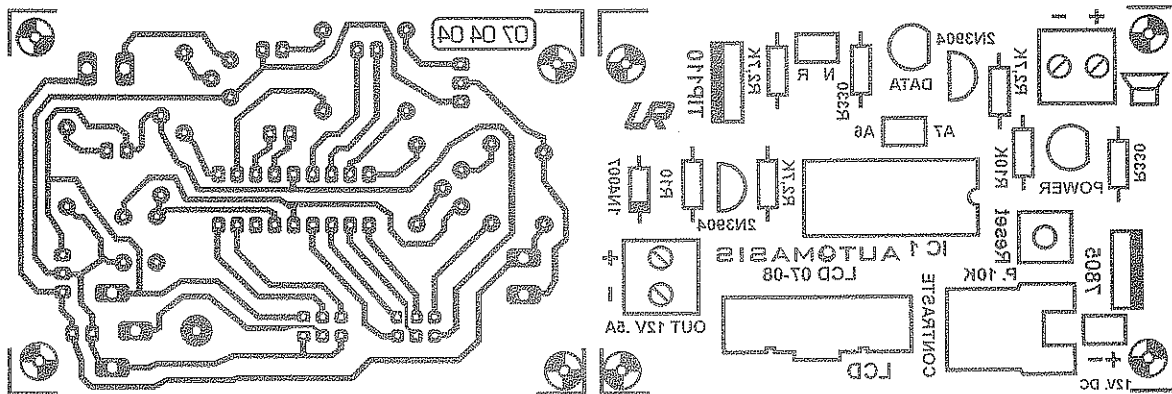


Figura 6.1.1. Diagrama de pistas y screen de elementos listos para la impresión láser correspondiente a la placa de la figura 5.5.1.8, pag 74.

Noten que el dibujo del lado derecho de la figura 6.1.1 está realizado un espejo, esto lo necesitamos por la transferencia térmica, en cuanto al tamaño de la placa deben considerar el chasis en donde van a colocar esta placa, como también donde deben ir los agujeros para los tornillos.

6.2 IMPRESIÓN DE LAS PISTAS Y SCREEN DE LOS ELEMENTOS.

Una vez que tenemos el diseño de nuestra placa, vamos a imprimirlo con una impresora láser o copiadora (que tengan los cartuchos toner de polvo en color negro), en un papel de transferencia térmica Press-n-Peel (o papel de transferencia PCB), que lo podemos conseguir en las tiendas electrónicas a un costo de más o menos 3 USD cada hoja.

Otra alternativa, la que yo más utilizo, y además se incluye en este libro es el **PAPEL FOTOGRÁFICO** tipo GLOSSY, de la empresa APLI, para el cual les doy los datos exactos:

APLI	Glossy Bright Paper	Ref. 04452	de 170g.	10 Und.
APLI	Glossy Bright Paper	Ref. 04135	de 170g.	50 Und.

Este papel fotográfico para impresoras de (inyección de tinta), vienen en cajas de 10 Und. o 50 Und. (Ref. 04135). El costo de la caja de 10 Und. es de 10 USD, y en algunas papelerías venden por unidades. Si no encuentran exactamente la misma hoja, pueden utilizar la hoja APLI de 125 g. de referencia 04451 de 10 Und. , o la de ref. 04134 de 50 Und., la única desventaja que tienen las hojas de 125 g. es que el papel se rompe al tratar de separar de la placa, pero sin embargo no es un problema se lo puede remojar toda la placa y así desprenderlo con la yema de los dedos.

No olviden que la impresión debe realizarse en una **IMPRESORA LÁSER** o fotocopidora y no por una impresora de inyección de tinta. Aquí está el truco, el papel fotográfico que hemos hablado anteriormente, sirve para imprimir fotografías específicamente en impresoras de inyección de tinta, al imprimirlo en una impresora láser, se da un efecto químico al unirse el toner de la impresora con la capa de barniz que tienen estas hojas, (lo mismo sucede con el papel Press-n-Peel), el hecho es que si se imprime con la de inyección a tinta, estas hojas ya no sirven.

Vamos a suponer que tenemos listo nuestra hoja con dicha impresión, el siguiente paso es aplicarle calor por el lado revés de las hojas y sobre las placas, para lo cual utilizaremos *la hoja de transferencia que viene con este libro* (UNIVERSAL PICmicro3), el calor de la plancha hace que el toner se derrita, y junto al barniz de la hoja se pegan en la lámina de cobre, para esto iremos explicando paso a paso todos los procedimientos necesarios, así como también pueden ver la secuencia fotográfica sobre como hacer placas PCB que contiene el CD en FabricaciónPCB\Ivisualizar.exe.

6.3 PREPARACIÓN DE LA PLACA (BAQUELITA O FIBRA DE VIDRIO).

MATERIALES.

- 1 placa de Baquelita o Fibra de vidrio, de una o dos caras de cobre
- 1 lija de metal Nro. 150
- 1 esponja de acero, de las utilizadas para lavar platos (lustre, estrella, etc.) que sea fina.

Primero debemos cortar el papel en dos partes de tal forma que quede solo la hoja que contiene el lado de las pistas, la otra parte es decir el screen de elementos lo utilizaremos casi al último.

Bien ahora debemos cortar la placa que puede ser de *Baquelita* o de *Fibra de vidrio*, esta última es la que más les recomiendo, ya que el acabado final es mucho mejor, además es más aislante y resistente a la humedad. Para las medidas del corte, debemos considerar 4 mm adicionales a cada lado de la placa en relación al del dibujo que vayamos a transferir, por lo que la medida de corte para nuestro grabador sería (69mm x 50mm). Utilizando una sierra de arco o una caladora con sierra de metal (debido a que este posee dientes más finos), cortamos la placa necesaria para nuestro grabador de micros.

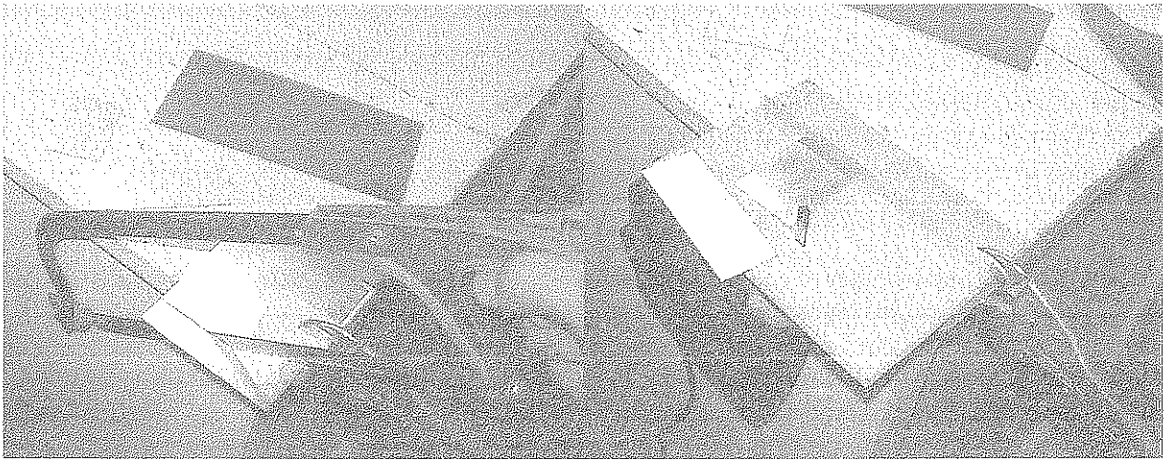


Figura 6.3.1. Paso 1. cortar la placa de 69 x 50 mm con una sierra o una caladora, esta última les permite cortes más rápidos y perfectos.

Como pueden observar yo utilizo la caladora, sujeta en una tabla y con una guía de aluminio, en la que regulo con unos tornillos y un alicate de presión la distancia de corte, consiguiendo igualdad en el tamaño de las placas para producciones en serie.

Una vez cortado la placa, debemos limpiar las limallas de cobre que quedan en los filos de la placa, con una fina lija de metal (Nro. 150), luego de esto debemos limpiarlo con una esponja de acero fina el lado del cobre donde vamos a transferir las pistas, notarán que la lámina de cobre cambia de color, esto porque estamos limpiando el óxido creado en la superficie y los rayones que pudiera tener.

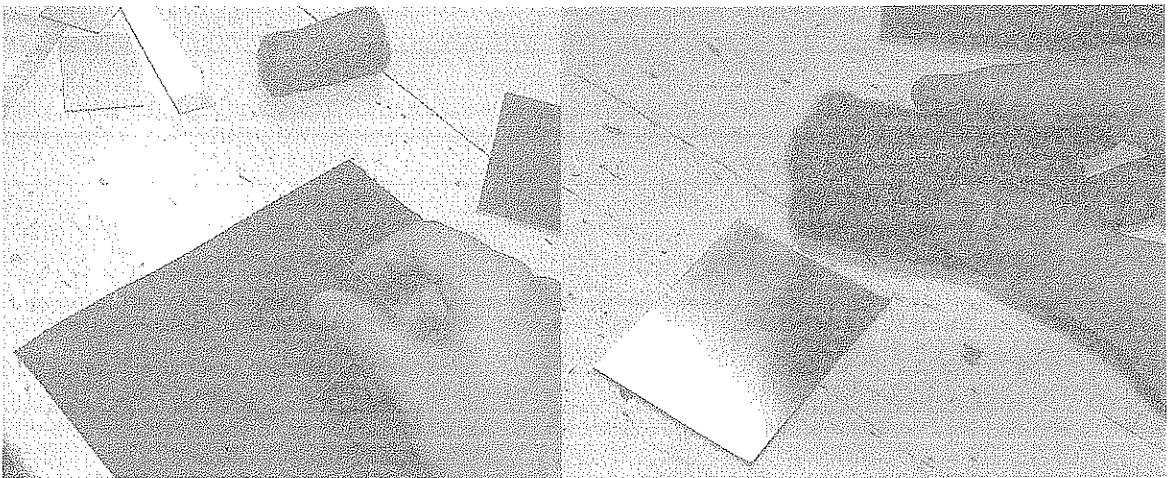


Figura 6.3.2. Paso 2. limpiamos los filos del corte realizado y la lámina de cobre oxidada.

Una recomendación muy importante, es que una vez limpia la placa, no se debe tocar con los dedos sobre la lámina de cobre, pues la grasa de los dedos genera óxido casi inmediatamente, si lo desea puede lavarlo posteriormente con crema lavaplatos del tipo arranca grasa.

Para el caso de querer guardar las placas se recomienda introducirla en una funda y cerrarlo con cinta adhesiva para así evitar el contacto directo con el aire.

6.4 TRANSFERENCIA TÉRMICA DEL PAPEL HACIA LA LÁMINA DE COBRE.

MATERIALES.

- 1 plancha domestica o una estampadora
- 2 pedazos de tela de calentador o franela
- La placa íntegramente limpia. (limpiar con esponja de acero)
- El papel con el diseño a transferir (Press-n-Peel o papel fotográfico Glossy)

Ahora vamos a realizar la transferencia propiamente dicha de las pistas sobre el lado de la lámina de cobre, recuerden esta debe estar completamente limpia y no debe ser tocada con los dedos, para manipularlo se debe sujetar por los bordes. Primero colocamos el papel fotográfico con el lado de la tinta sobre el lado del cobre, sin moverlo mucho introducimos debajo de la tela, todo esto sobre una mesa rígida y luego pasamos la plancha que debe estar al máximo de la temperatura, le aplicamos presión con todo el peso de nuestro cuerpo, por alrededor de 20 a 30 segundos, luego de esto lo retiramos e inmediatamente lo colocamos en otra parte de la mesa que

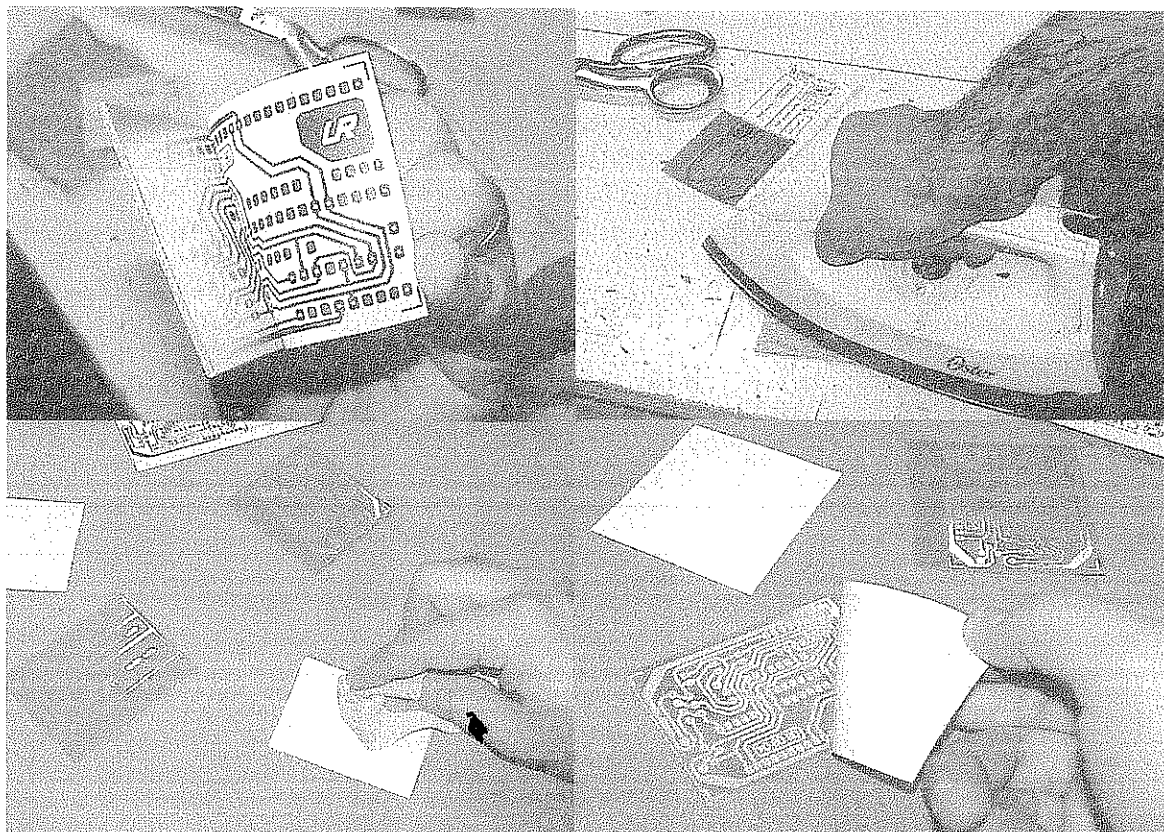


Figura 6.4.1. Paso 3. Aplicamos presión con una plancha bien caliente por 30 seg. y lo dejamos enfriar presionando con otro trapo, para luego retirar el papel cuidadosamente.

se encuentre fría, y con otro trapo le aplicamos presión uniforme frotándolo de un lado a otro hasta que este se enfríe, con la finalidad de que toda la tinta (toner + barniz) se pegue a la lámina de cobre y así poder retirar el papel sin que se presente partes cortadas o faltantes.

Ahora les explico razones por la que podría salir mal la transferencia térmica, si la plancha no es suficientemente caliente, gran parte de la tinta no se pegará a la placa, en este caso utilice otra plancha, deben notar que el papel se amarilla un poco por efecto del calor, otra razón puede ser también que estén utilizando una tela muy gruesa, cambie a otro tipo de tela, y por último puede ser la poca presión aplicada con la plancha, se debe prácticamente apoyarse sobre la plancha y frotarlo sobre toda la superficie de la placa.

Una solución muy eficaz es utilizar las planchas estampadoras o fusionadoras, estas son utilizadas para estampar camisetas o forros pegables, tienen un lado de caucho resistente al calor y su presión es muy alta, así como la temperatura que puede entregar es de hasta 500°C, (nosotros utilizaremos de 350 a 400°C), estas planchas tienen una superficie de 40 x 40 cm, ideal para placas de 30 x 20 cm, ya que la plancha doméstica es ideal para placas de 10 x 20 cm.

6.5 PROCESO DE ATACADO (REDUCCIÓN) DEL COBRE.

MATERIALES.

- 1 recipiente de plástico, (no metálico)
- ½ vaso con agua tibia
- 1 palillo de pincho o una pinza de plástico
- 1 funda de cloruro férrico (en polvo)
- 1 calentador de agua para peceras (opcional)

Para reducir el cobre sobrante, es decir el que no está protegido por la tinta y el barniz, necesitamos preparar un atacador, existen 2 tipos de atacadores: los rápidos y los lentos, los rápidos como por ejemplo la combinación de 50 ml de ácido clorhídrico y 50 ml de agua oxigenada, pueden reducir el cobre no protegido al cabo de unos pocos segundos, pero tiene la desventaja de ser difíciles de conseguir en el mercado, los lentos en cambio como el Cloruro férrico se lo encuentra en cualquier tienda electrónica pero el proceso de atacado podría tomar hasta 1 hora. Sin embargo por ser menos agresivo y porque no emana muchos gases tóxicos, utilizaremos el cloruro férrico.

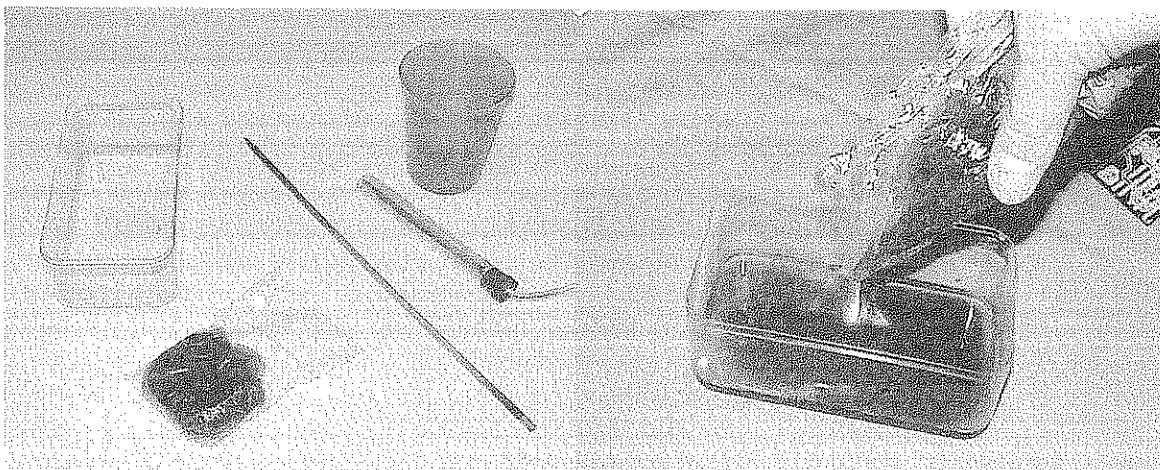


Figura 6.5.1. Materiales a utilizar, y a la derecha mezcla del cloruro férrico con el agua.

Para su preparación primero se recomienda un lugar con buena ventilación, no utilizar reloj con pulseras de metal, pues el cloruro férrico ataca a los metales, también tome en cuenta que al contacto con la ropa o la piel, produce una mancha amarillenta, por lo que debe tomar las precauciones necesarias, y en caso de darse contacto con la piel debe lavarse con agua y jabón.

Una vez que estemos listos procedemos a preparar la solución ácida, primero colocamos el $\frac{1}{2}$ vaso de agua tibia en el recipiente de plástico, luego colocamos poco a poco si es posible con una cuchara de plástico, todo el contenido de la funda de cloruro férrico, es normal que el agua se empiece a calentar (debido a la reacción química), el palillo de pincho lo utilizaremos para revolver el líquido y así ayudar a disolver el cloruro férrico.

PELIGRO: Nunca ponga todo el cloruro férrico de una sola vez sobre el agua, la reacción muy brusca podría hacer explotar y producir quemaduras en la ropa y en la piel.

Esta solución ya preparada, puede ser almacenada en un envase de plástico o vidrio para luego ser utilizada varias veces, hasta que el ácido se contamine tanto que ya no sea posible corroer placas (debido a que el efecto es cada vez más lento), para entonces se debe desechar.

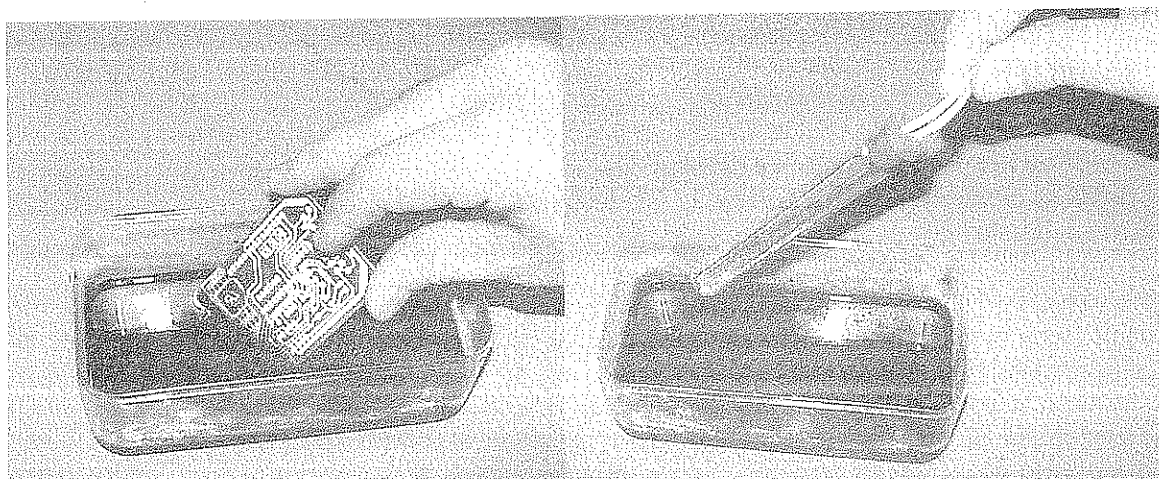


Figura 6.5.2. Introduzca la placa virgen en la solución ácida, si el ácido ya ha sido utilizado varias veces y tiene poca acción, introduzca el calentador para peceras.

Procedemos a introducir la placa del grabador de PIC'S en la solución ya preparada, el tiempo de corrosión por ser la primera vez, puede variar entre 20 y 30 minutos, por lo que debe revisar de vez en cuando si el cobre no protegido ha sido eliminado utilizando el palo de pincho. También podemos ayudar a que el proceso sea más rápido, moviendo el agua de un lado a otro, esto permite que el cobre disuelto, por efecto de la corriente generada por el movimiento, se deposite en el fondo del envase, así permitimos que la lámina tenga contacto directo con el ácido.

En ocasiones especiales cuando por ejemplo la solución ya ha sido utilizada varias veces, el proceso de corrosión es mucho más lento, el calor y el movimiento pueden ayudar a reaccionar al ácido, para esto necesitamos un envase vertical, introducimos en el fondo una generador de oxígeno, del utilizado en las peceras, también introducimos un calentador de peceras de 2 Gls., el ascenso de las burbujas genera movimiento en el agua, que junto al calor generado por el calentador de pecera, ayudan a corroer más rápido el cobre de las placas, en este caso la placa se coloca verticalmente, permitiendo que el cobre disuelto caiga rápidamente al fondo del envase.

6.6 PROCESO DE LIMPIEZA DE LA PLACA YA ATACADA CON ÁCIDO.

Una vez que el ácido terminó de eliminar el cobre expuesto, retiramos la placa del ácido y lo lavamos con abundante agua del grifo, las pistas, pads, etc., en esta fase se ven de color negro, esto se debe a que el papel se encuentra remojado, pero cuando se seca vuelve a ser de color blanco, ahora nos resta limpiar todo el residuo de papel y tinta de 2 maneras posibles, la primera es utilizando thinner, acetona, o cualquier disolvente, pero esto tiene un efecto secundario si bien limpia las pistas, un poco de tinta negra se impregna en algunas partes de la placa, dando la apariencia de sucia, por tal razón yo opté por limpiar con la misma esponja de acero y un poco de agua, el resultado de la limpieza no deja rastro de tinta y se ve muy nítido.

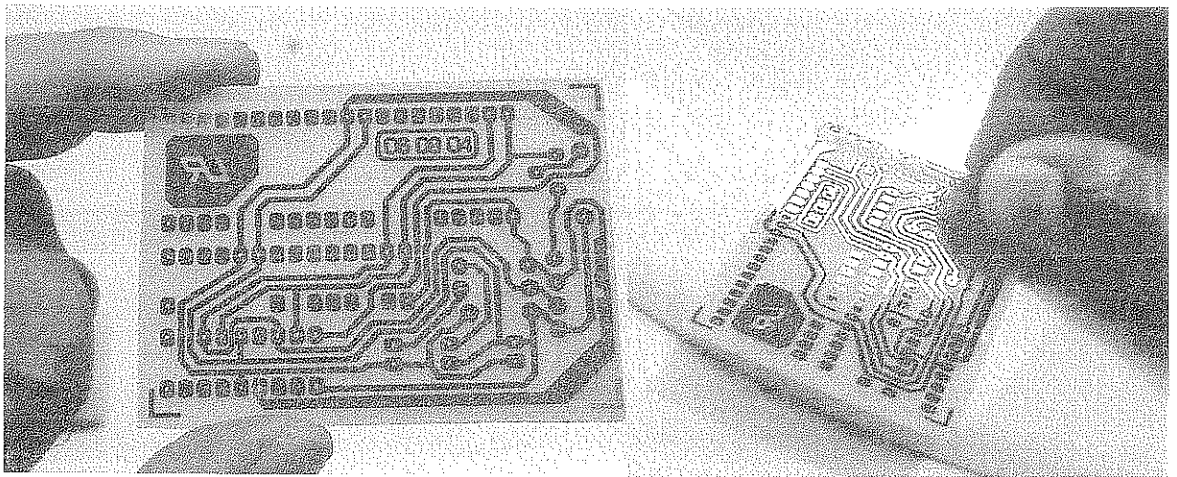


Figura 6.6.1. Placa recién sacada del ácido, y limpieza de la tinta con esponja de acero y agua.

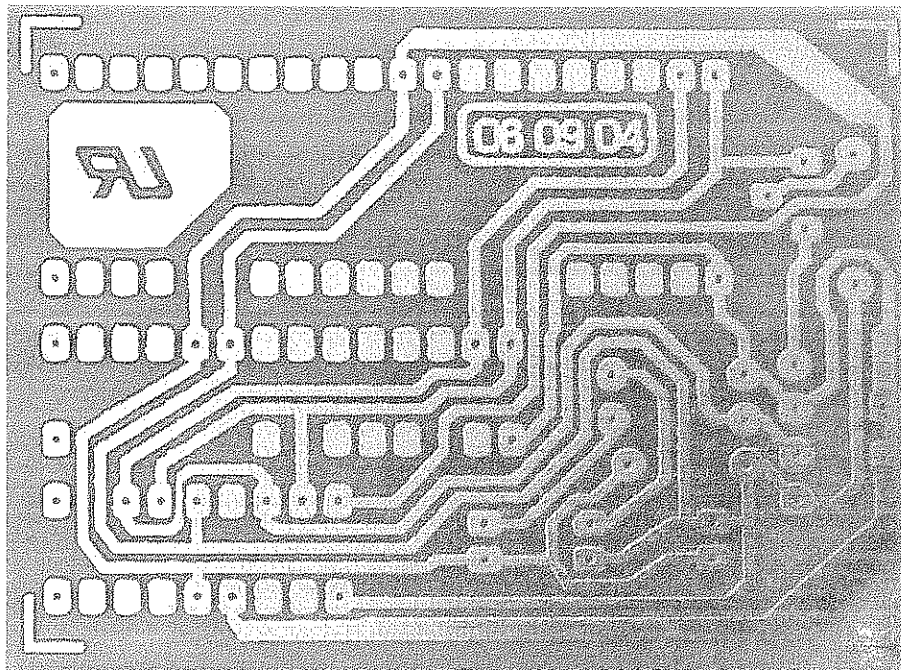


Figura 6.6.2. Apariencia que debe tener la placa ya atacada y libre de tinta (toner + barniz).

6.7 TRANSFERENCIA TÉRMICA DEL SCREEN DE LOS ELEMENTOS.

El screen de elementos, no es nada más que textos, información, datos y figuras que indican el lugar donde se debe insertar los elementos electrónicos, son muy útiles ya que evitan que la persona se equivoque de lugar al insertar una resistencia, o coloque al revés un diodo, etc, el método de transferirlo es el mismo que hicimos para la transferencia de las pistas, este se debe colocar a la misma altura y posición que están las pistas y también se debe tener cuidado de no colocar al revés (recuerde que este papel es espejado). Para que se adhiera bien, en el caso de placas de baquelita, se debe lijar bien el lado donde se va a colocar el screen, ya que tienen una capa de laca o barniz, para el caso de placas de doble cara, al disolverse la cara que no se necesitaba, esta queda bien limpia y porosa, por lo que no hace falta limpiarle.

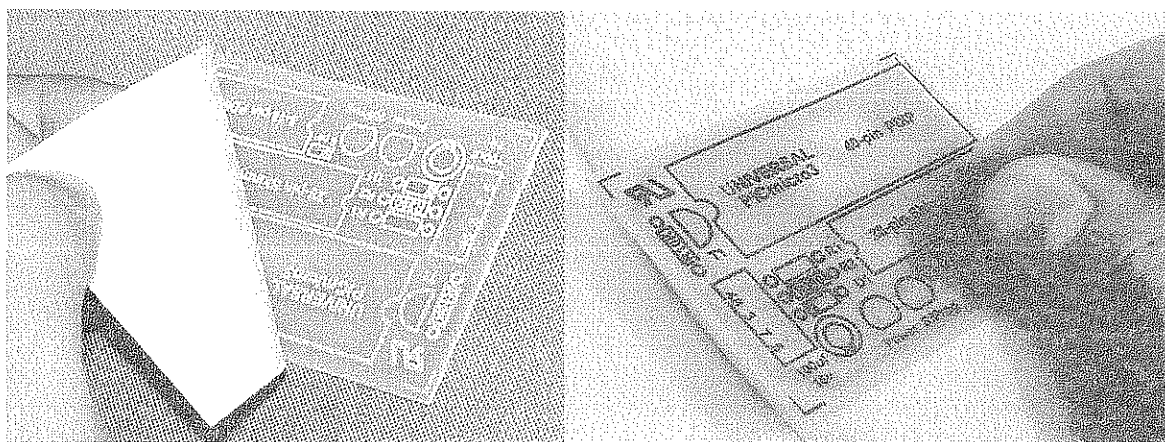


Figura 6.7.1. Retiramos el papel después de haber hecho la transferencia térmica, y limpiamos el residuo de papel con una esponja de acero y agua, teniendo cuidado de no sacar la tinta.

Una vez que se retiró el papel, notarán que el texto es blanco, en este caso debemos limpiarlo suavemente con una esponja de acero y agua todo el residuo de papel, teniendo cuidado de no borrar el screen, debemos dejarlo secar y ver que partes se volvieron a blanquear, para nuevamente lijar con la esponja de acero estos sectores.

Para el caso de las personas que hicieron el screen con papel Press-n-peel, no hace falta estar limpiando, ya que estas no dejan residuo de papel, sino una fina capa de material filmico de color azul.

6.8 PERFORACIÓN DE LA PLACA.

MATERIALES.

- 1 broca para metal, de 1mm de diámetro
- 1 taladro o moto-tool (taladro miniatura)
- 1 pedestal para taladro (opcional).

Lo único que nos falta para que la placa esté lista es realizar los respectivos agujeros, para el caso de nuestro grabador de PIC'S, necesitamos hacer 48 agujeros con broca de 1 mm. lo ideal es disponer de un moto-tool o taladro miniatura, ya que estos son de fácil manipulación, además pueden soportar brocas de 0,3 mm en adelante, para este caso se debe hacer una hendidura con un

punzón y un martillo en cada lugar donde se va hacer un agujero, con la finalidad de que sirvan de guía para la broca.

Yo utilizo un taladro cuyo mandril puede soportar brocas desde 0,5 mm hasta 10 mm, y dispongo de un pedestal, en este caso no necesito hacer las hendiduras con punzón, ya que la perforación se realiza completamente perpendicular a la placa y sin que se me desvíen las brocas esto gracias a la ayuda del pedestal.

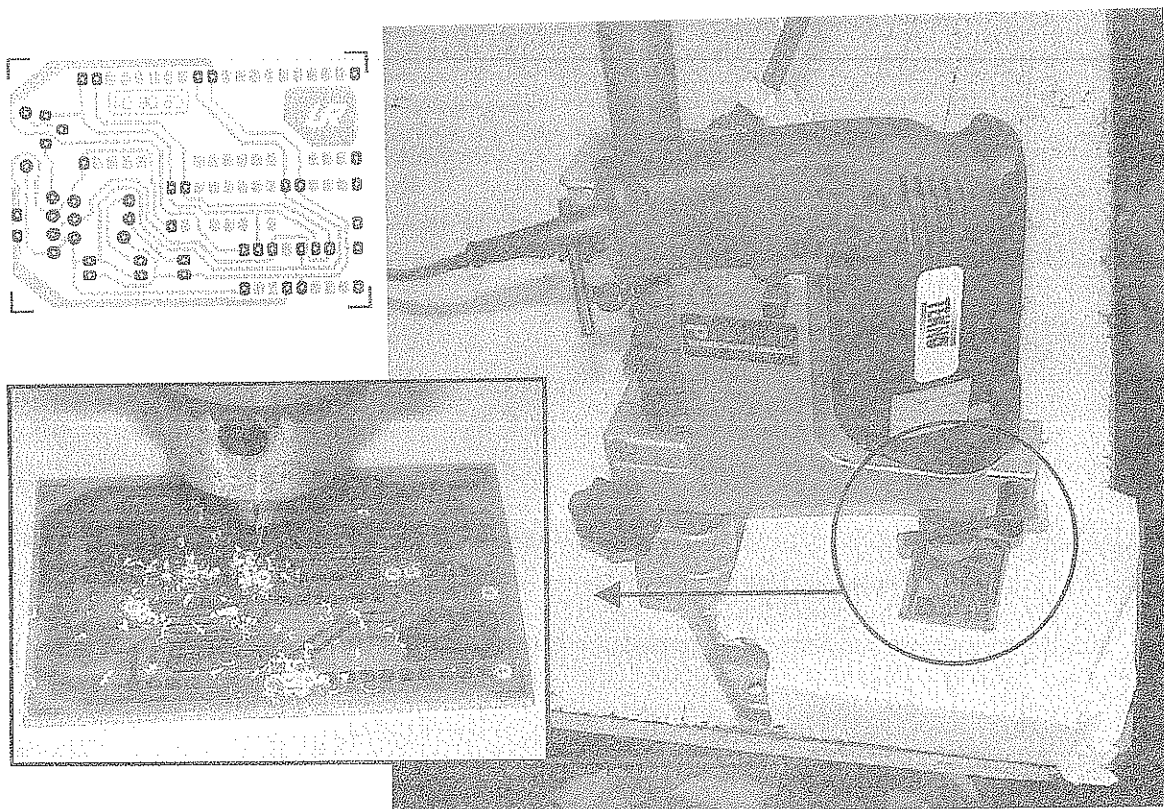


Figura 6.8.1. Pedestal y taladro con broca de 1 mm. para perforar la placa PCB.

6.9 SOLDADURA DE ELEMENTOS.

Los materiales que necesitamos para nuestro grabador de PIC'S, son los siguientes:

MATERIALES.

- a) Un metro de cable de 4 hilos multifilar
- b) Un conector DB9 hembra con su respectivo cajetín
- c) Dos leds de 5mm, un rojo y un verde
- d) Un diodo zener de 5.1 V. a ½ vatio
- e) Una resistencia de 390 Ω a ¼ de vatio naranja-blanco-café
- f) Una resistencia de 1 K Ω a ¼ de vatio café-negro-rojo
- g) Una resistencia de 10 K Ω a ¼ de vatio café-negro-naranja
- h) Un capacitor electrolítico de 100 μ F/25V.
- i) Un transistor NPN 2N3904
- j) Tres zócalos, uno de 18 pines, otro de 28 pines y otro de 40 pines.

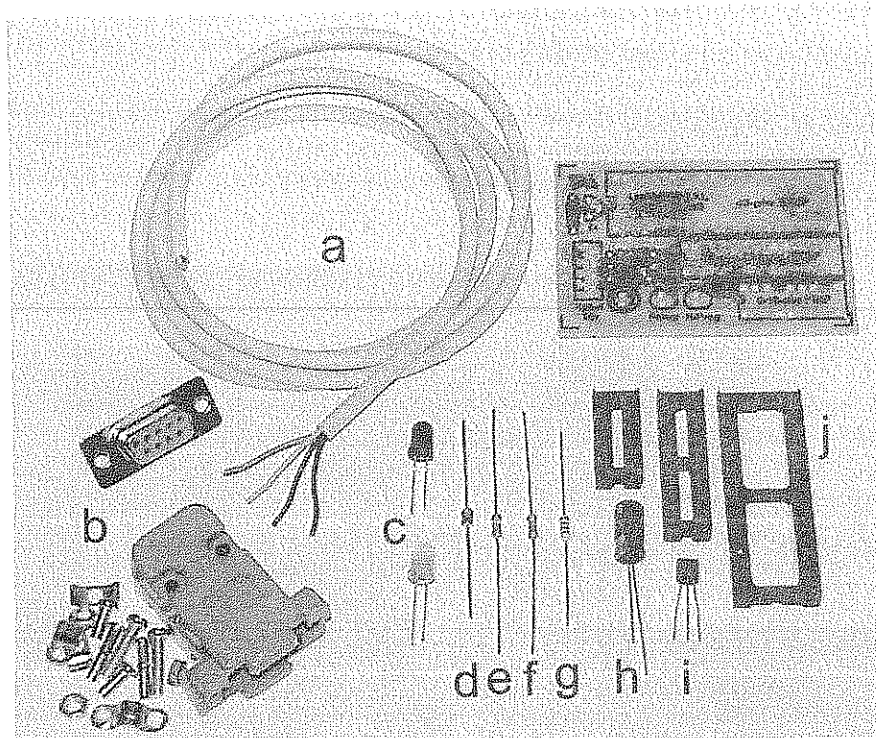


Figura 6.9.1. Materiales para la fabricación del grabador de PIC'S UNIVERSAL PICmicro3.

Las herramientas y materiales que todo soldador electrónico debe tener son los siguientes:

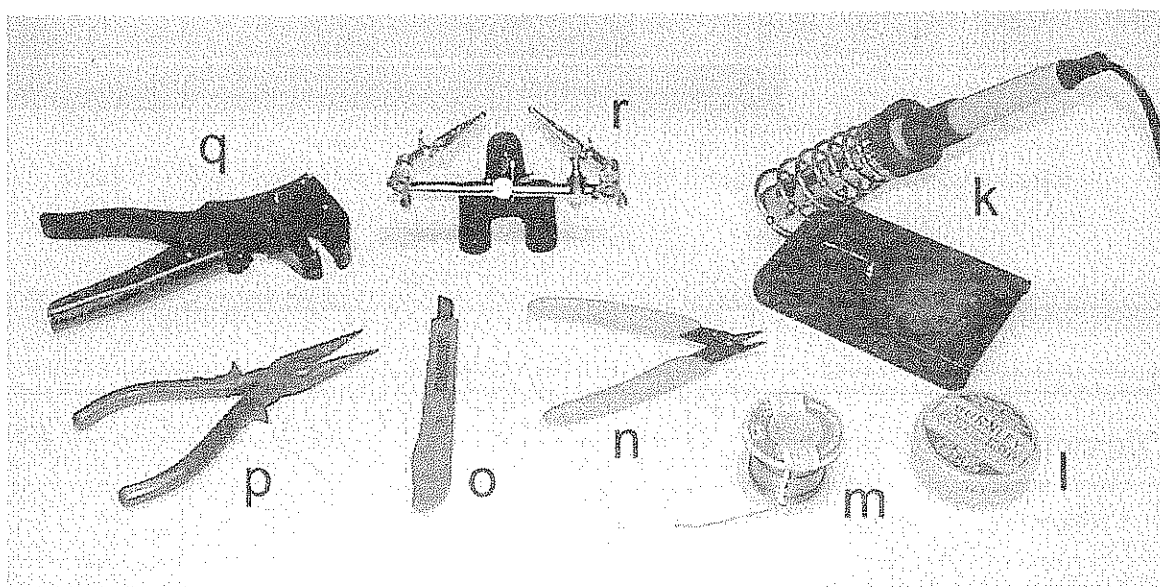


Figura 6.9.2. Herramientas y materiales que se debe disponer para procesos de solda:
PRINCIPALES : k) cautín tipo lápiz de 20W a 30W con su respectivo soporte, l) pasta de soldar, m) alambre de solda de 1 mm de diámetro de estaño(60%) y plomo(40%), con centro de resina n) pinza de corte, o) estilete o bisturí, p) alicate de punta.
OPCIONALES: q) pinza pelacables, r) sujetador de placas o una entenalla pequeña.

Una vez que tenemos listos los materiales y herramientas, empezamos por los zócalos, estos debemos prepararlos sacando algunos pines que no necesitamos, con el alicate de punta empujamos uno por uno los pines no necesarios como muestra la siguiente figura:

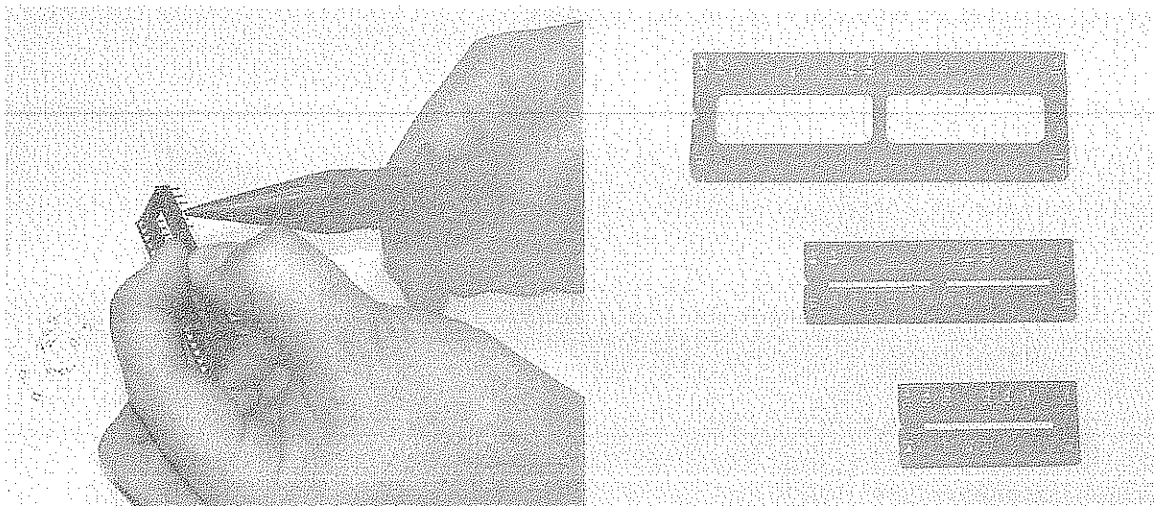


Figura 6.9.3. Retiramos los pines que no necesitamos, empujando uno por uno con la pinza de punta, hasta que queden como el de la fotografía de la derecha.

Las resistencias y el diodo, debemos doblarlos, a la medida necesaria y con la ayuda de la pinza de punta:

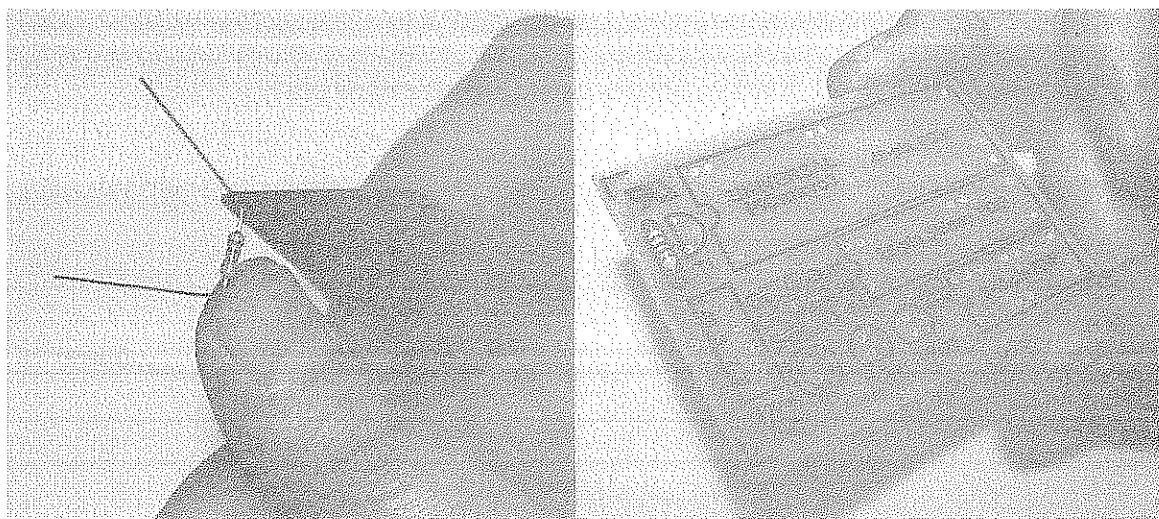


Figura 6.9.4. Doblar las resistencias y el diodo, de acuerdo a la distancia de las perforaciones, en el momento de insertar el diodo zener se debe tomar en cuenta la polaridad.

Se debe seguir en secuencia la soldadura de los componentes, primero los elementos más bajos y luego los más altos como el capacitor, de esta manera el orden para ir soldando sería: resistencias, diodo, zócalos, transistor, leds, y por último el capacitor, luego de todo esto soldamos el cable que daremos las indicaciones posteriormente.

Iniciamos con la suelda de las resistencias y el diodo, estos debemos insertarlos, y luego doblamos las patitas hacia el exterior, con la finalidad de que al dar la vuelta la placa para soldar, estos no se caigan, luego de esto colocamos en el soporte para placas y procedemos a soldar, el mejor método de suelda, es calentar un poco el elemento a soldar y luego poner el estaño, movemos la punta del cautín de arriba abajo, tocando el alambre de suelda y el elemento, esto permite una rápida adherencia y una buena soldadura.

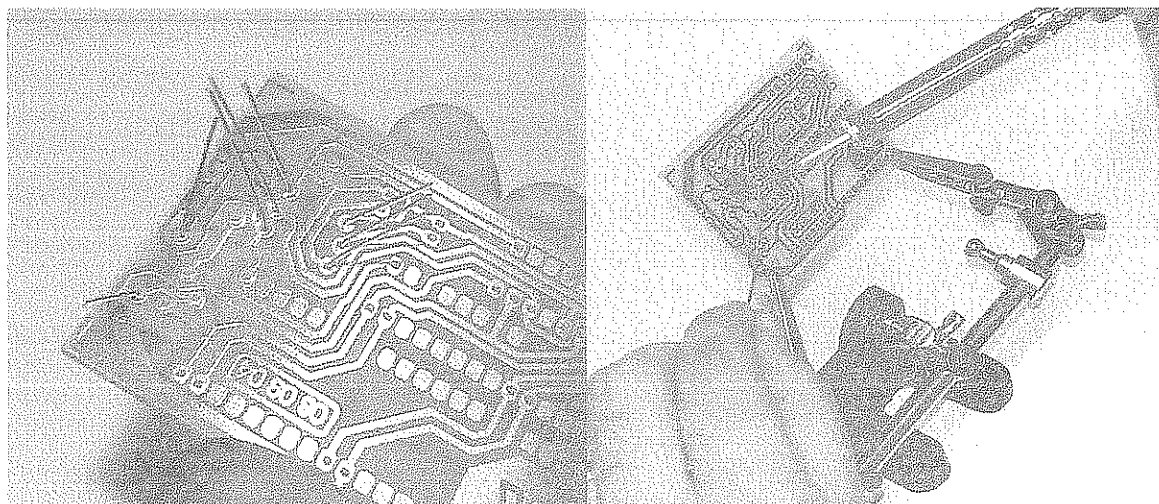


Figura 6.9.5. Doblar las patitas de las resistencias hacia el exterior, sujetarlo en el sujeta placas y proceder a soldar.

Si usted no dispone de un soporte para placas, puede soldar de la siguiente manera: con la uña de su dedo índice sujete la resistencia. El alambre de suelda colóquelo al filo de una mesa y con su otra mano manipule el cautín, como lo muestra las siguientes fotografías:



Figura 6.9.6. Sujete la resistencia con la uña, coloque el alambre de soldar al filo de una mesa y sin soltarlo empiece a soldar.

Los leds deben ser colocados de la siguiente manera, donde dice Power va el led rojo y donde dice R/Prog, va el led verde, coloque correctamente la polaridad de los leds, así mismo asegúrese de colocar el capacitor de acuerdo a la polaridad que le indica el screen. Una vez que ha terminado de soldar los elementos, la placa debe tener la siguiente apariencia:

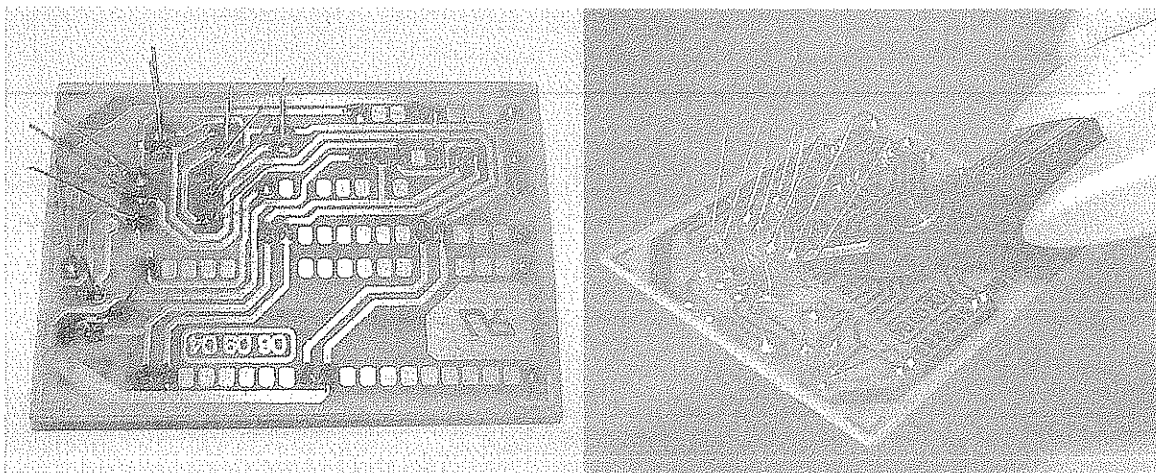


Figura 6.9.7. Una vez soldado los elementos, corte los alambres sobrantes.

Ahora con la pinza de corte, cortamos todos los alambres que sobresalen de la parte posterior, no olvide guardar el alambre que corte del diodo zener, este nos servirá luego para sujetar el cable de 4 hilos.

El cable de conexión al PC lo preparamos de la siguiente manera: primero retire la envoltura plástica del alambre, más o menos unos 2 cm, esto con la ayuda del estilete si lo prefiere.

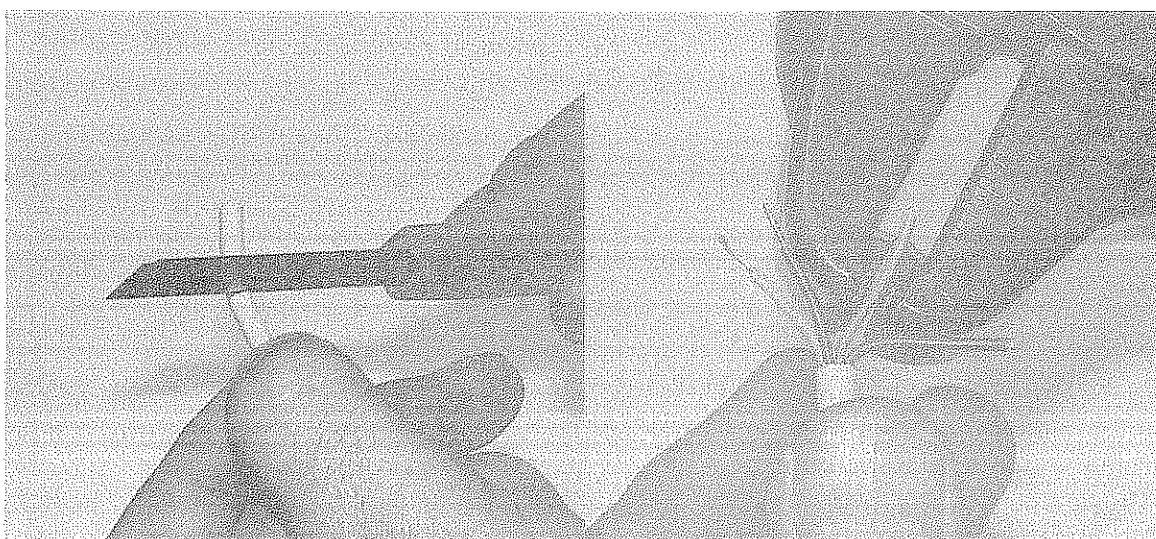


Figura 6.9.8. Pele los alambres con la ayuda de un estilete o un pelacables, luego debe torcer los filamentos para que permanezcan uniforme.

Una vez pelado el cable, introduzca las puntas de los alambres en la pasta de soldar, y con la ayuda del cautín estáñelo todos los alambres como muestra las siguientes fotografías:

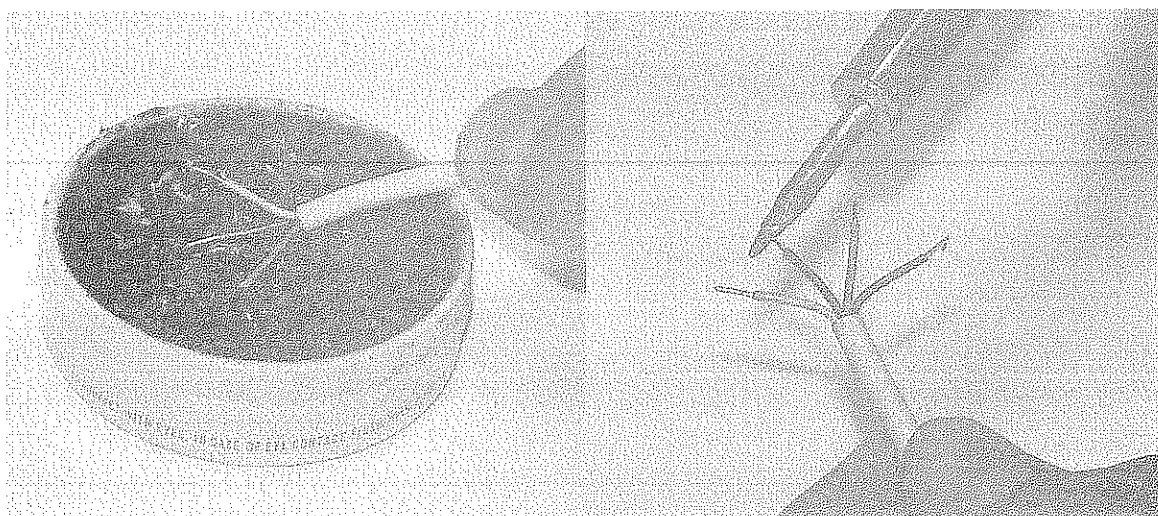


Figura 6.9.9. Introduzca las puntas del alambre en la pasta de soldar y luego estáñelo con el cautín previamente cargado de suelda.

Con las puntas de los alambres estañados, los filamentos permanecen juntos, ahora introduzca en las perforaciones de la placa y sosténgalo con sus dedos hasta que logre soldarlos.



Figura 6.9.10. Introduzca las 4 puntas de los alambres y sosténgalo con su dedo, luego proceda a soldar como se aprendió en los casos anteriores.

Una vez soldado los alambres procedamos a sujetarlo a la placa, para que el movimiento no los rompa, para esto necesitamos empujar el alambre del otro extremo hasta que la envoltura recorra

por lo menos 1 cm, lo suficiente para poder ser remordido con el alambre que se debe haber guardado al cortar el diodo zener, este debe doblarse formando un arco y debe atrapar al cable con todo su envoltura (ver Fig. 6.9.11 y 6.9.15.), luego de doblarlo suéldelo a la placa tratando de no calentar mucho al alambre, porque este podría derretir la envoltura del cable multifilar.

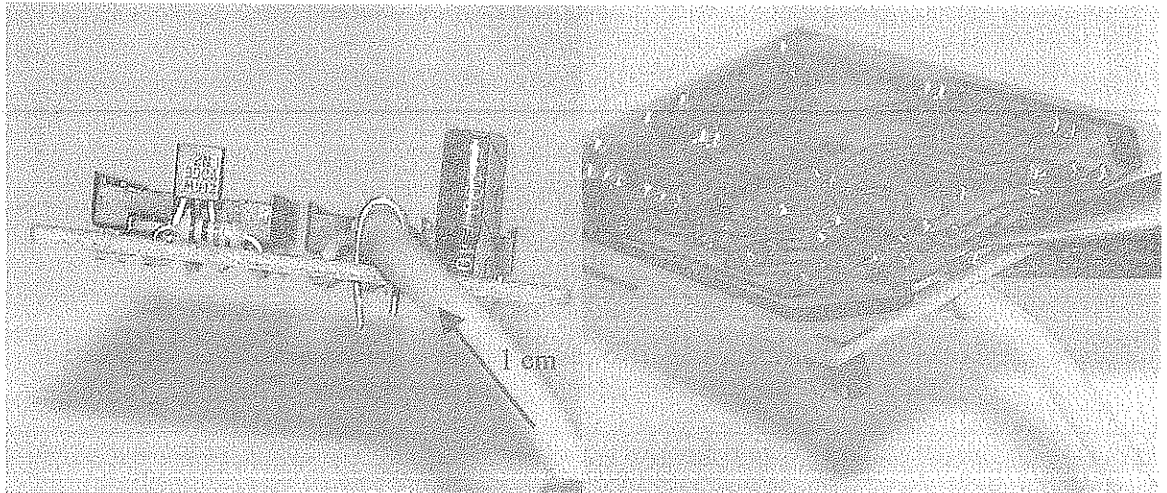


Figura 6.9.11. Introduzca el alambre (residuo del diodo zener) en las 2 perforaciones, y con la ayuda del alicate de punta doble como ilustra la fotografía de la derecha.

En el otro extremo del cable se debe soldar el conector DB9, que nos servirá para la conexión del puerto serial del computador, primero es necesario sujetarlo el conector en un lugar fijo, para tener ambas manos disponibles para la suelta, estañamos los terminales 3,4,5,7,y 8 y luego procedemos a soldar cada uno de los cables previamente estañados como se aprendió en la figura 6.9.9. Se debe tener en cuenta el lugar que le corresponde a cada cable, para esto debe fijarse como soldó en la placa y si por ejemplo el cable café está en el Nro. 3 pues deberá soldar en el pin 3 del DB9, el cable que soldó en el Nro. 4 8, debe soldarse con un puente al pin 4 y 8 (ver Figura 6.9.13).

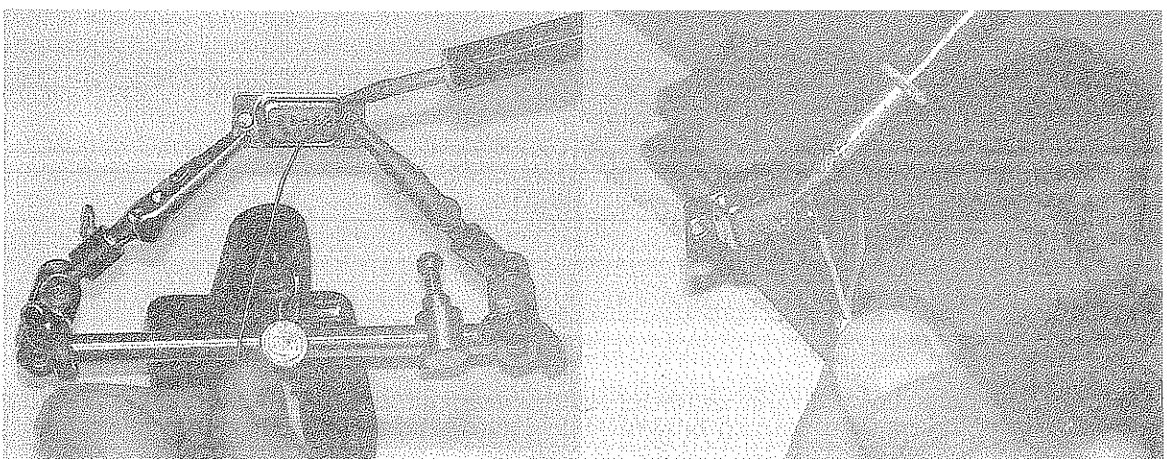


Figura 6.9.12. Sujete el conector DB9 y estañe los terminales, si no dispone de un sujetador, podría servirle una entenalla pequeña, es importante que disponga de sus 2 manos para soldar.

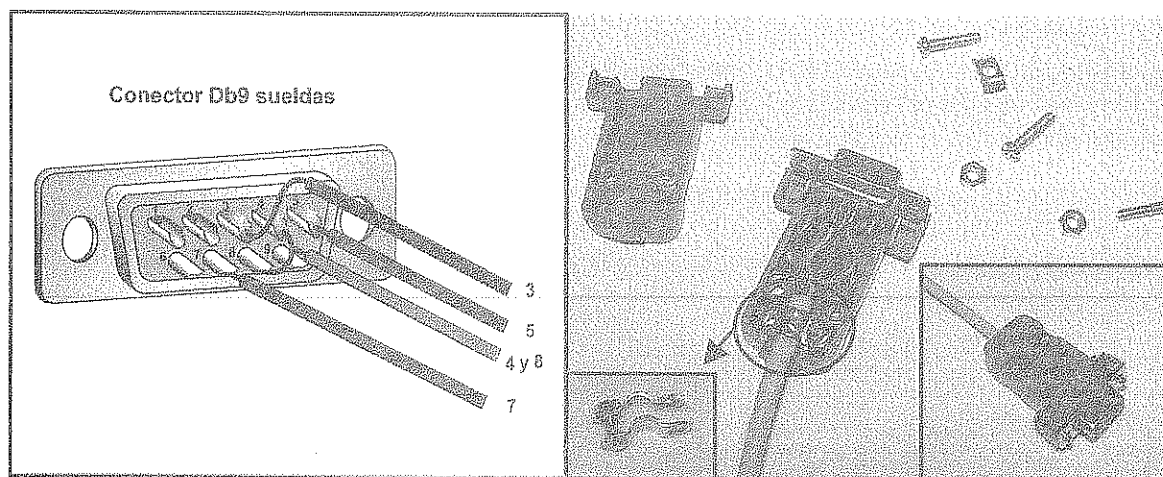


Figura 6.9.13. Suelde cada cable en el lugar que le indica el screen de la placa UNIVERSAL PICmicro3, luego coloque el cajetín del conector DB9.

El puente que une el pin 4 y 8 puede hacerlo con el mismo alambre, soldando primero el un pin y luego doblando hasta alcanzar el otro pin. Una vez colocado los cables dentro del cajetín, puede colocar un poco de silicón con la pistola térmica, esto lo ayudará a que no se rompa con la manipulación del conector.

Para limpiar los residuos de suelda (pasta) que se encuentra en las pistas de la placa, podemos utilizar un cepillo de dientes que ya no se utilice, introducimos las cerdas en un poco de thinner, y cepillamos la placa teniendo cuidado de que el thinner no se derrame por el lado posterior de la placa, es decir el screen ya que podría borrarlo.

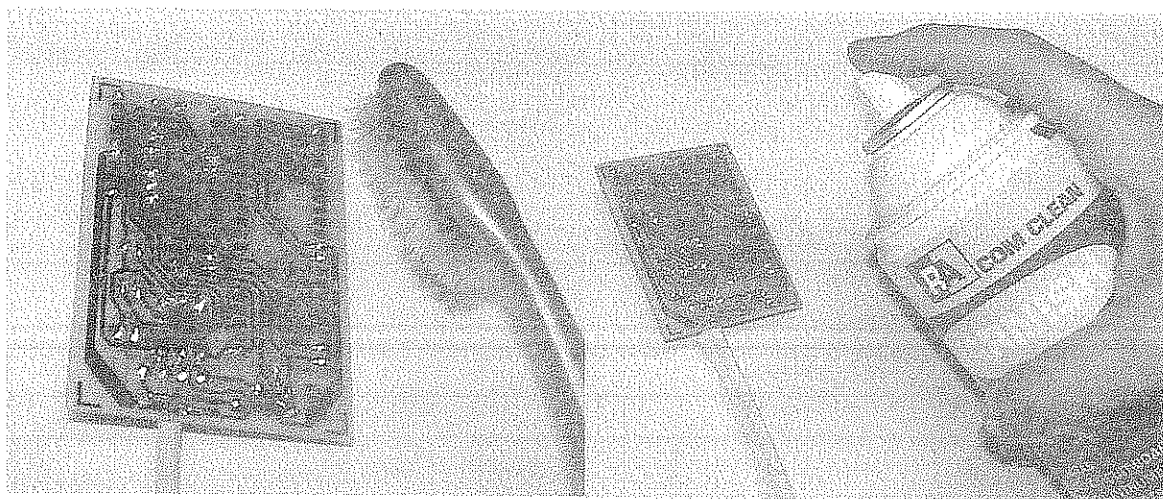


Figura 6.9.14. Limpie la placa con cuidado mucho cuidado, ya que el thinner podría borrar el screen, déjelo secar y posteriormente puede darle una capa de laca o barniz.

Para evitar que las pistas de cobre se oxiden, se debe dar una capa de barniz en spray para circuito impreso, este también lo utilizan para cubrir el rebobinado de los motores, la marca que conozco es RA com clean VARNISH, este le dará un acabado brillante transparente, pero si desea darle

un acabado más profesional, (ver CD:\Fotos libro\grabador PIC2), puede utilizar laca transparente con un poco de pintura verde, esto se debe hacer preparar en las tiendas que venden y preparan pintura automotriz, se debe llevar una placa de circuito impreso que tenga la máscara antisoldadura de color verde, se le pide que saquen el color de la placa mezclando laca transparente con pintura automotriz. Una vez que el color sea el correcto, y con la ayuda de un compresor y la pistola de pintar se procede a rociar el lado de las pistas, teniendo cuidado de no pintar el cable, para esto se debe cubrir con un poco de cinta adhesiva.

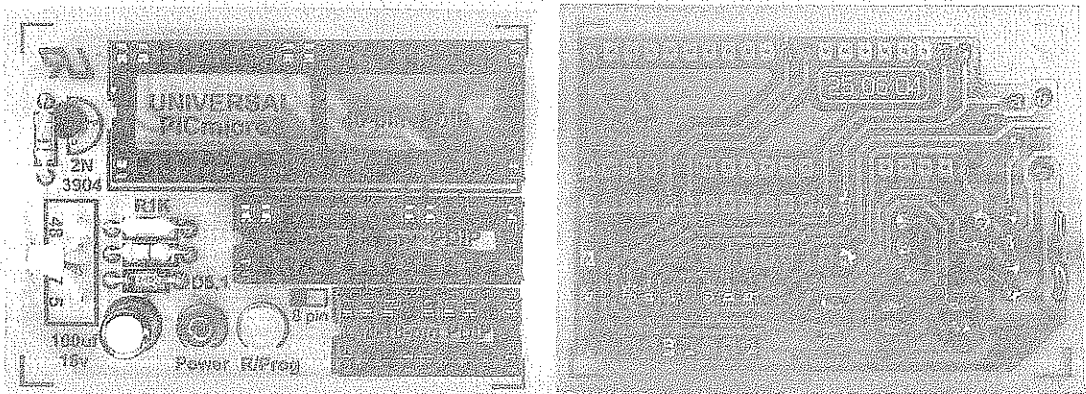


Figura 6.9.15. Apariencia del grabador de PIC'S con la máscara antisoldadura de color verde.

6.10 CHASIS O CAJA PARA PROYECTOS.

En este subcapítulo se pretende dar ideas muy prácticas de cómo construir un chasis o caja para proyectos, primero debemos buscar en el mercado las cajas que podrían servirnos para contener un circuito, una de ellas es la caja DEXON, que se utilizan para colocar tomacorrientes externos.

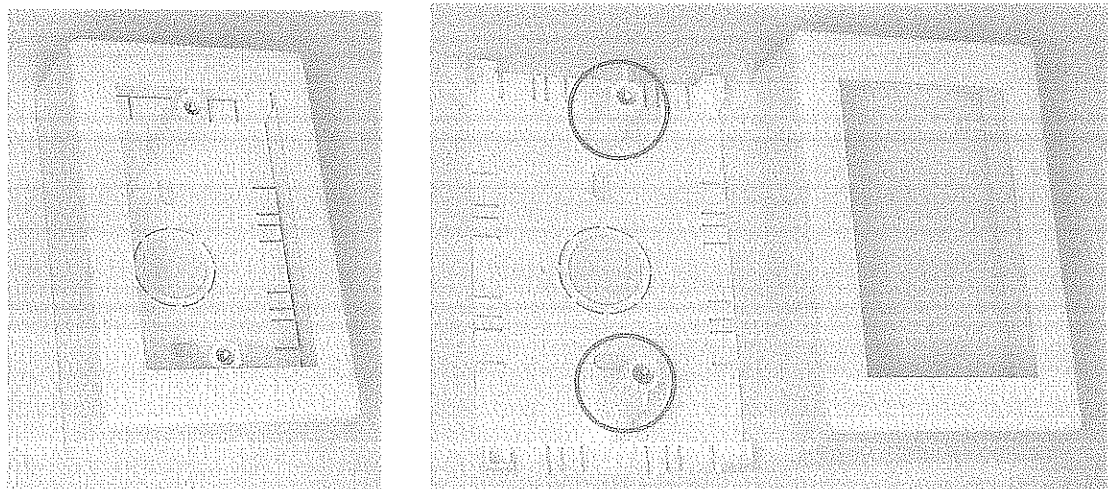


Figura 6.10.1. La caja para tomacorrientes de la marca DEXON, es ideal para proyectos.

Estas cajas también son utilizadas para instalaciones de cableado estructurado (ver Figura 5.7.4.2.), los cajetines telefónicos también nos podrían servir, las cajas para breakers, y si no nos sirven ninguna de estas pues por último mandamos a doblar una caja a la medida.

A continuación mostramos como hacer un teclado para el control de accesos, primero utilizamos la caja DEXON o cualquier otra marca, cortamos las 2 tuercas y lo colocamos en otro lugar, utilizando pega instantánea.

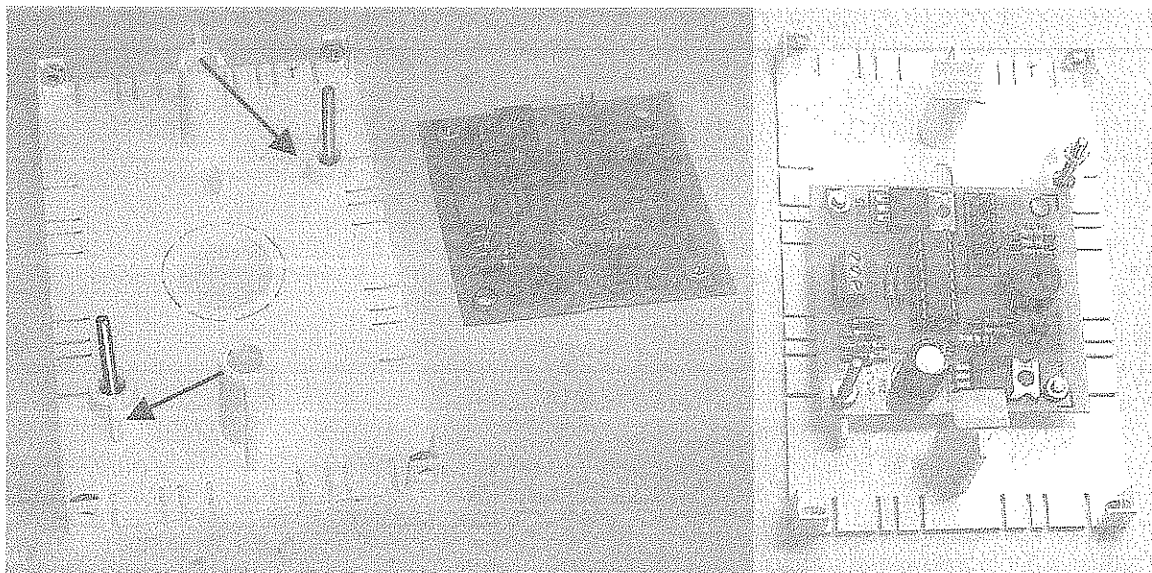


Figura 6.10.2. Cortamos las tuercas y las pegamos en cualquier lugar que deseemos.

Para fabricar el panel del teclado, primero dibujamos en un programa de dibujo todas las teclas, luego mandamos a imprimir en acetato con una impresora láser a color, pegamos un adhesivo blanco por la parte posterior, con la finalidad de que las partes transparentes se vean blancas.

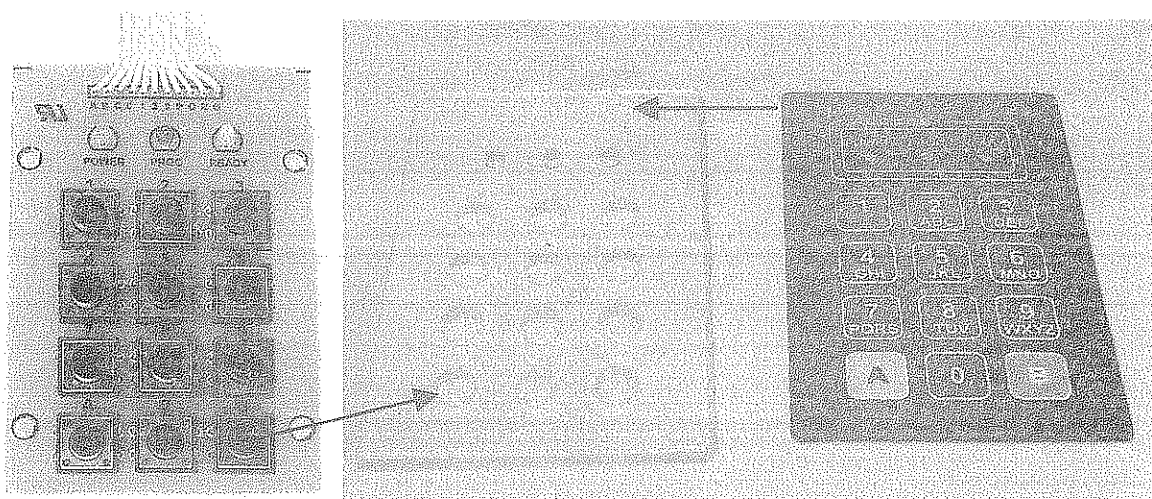


Figura 6.10.3. Cortamos una lámina de acrílico y lo practicamos los agujeros por donde pasaran las teclas, imprimimos en una lámina de transparencia (acetato) los dibujos de las teclas y lo pegamos sobre el acrílico.

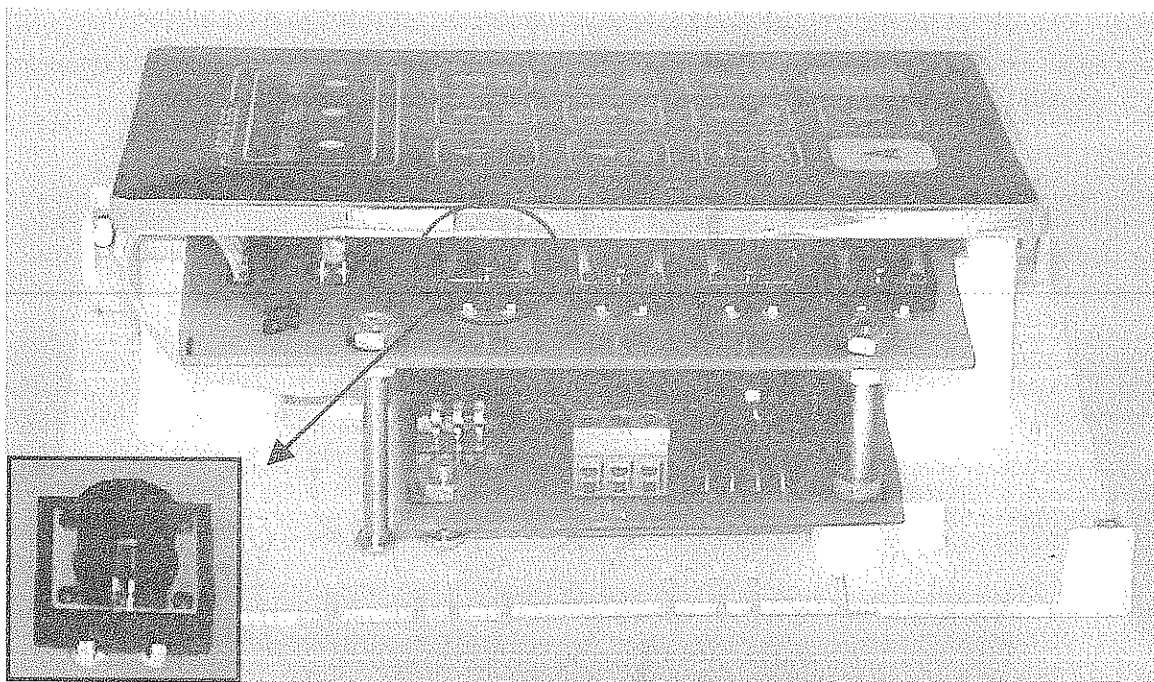


Figura 6.10.4. Fotografía lateral del teclado para control de accesos, noten que los pulsadores son de 4,5 mm de alto suficiente para que atraviese a la lámina de acrílico que mide 4mm. de espesor

En el mercado se puede encontrar diferentes cajas metálicas o plásticas, para proyectos o para instalaciones eléctricas, aquí una fotografía de 2 de ellas.

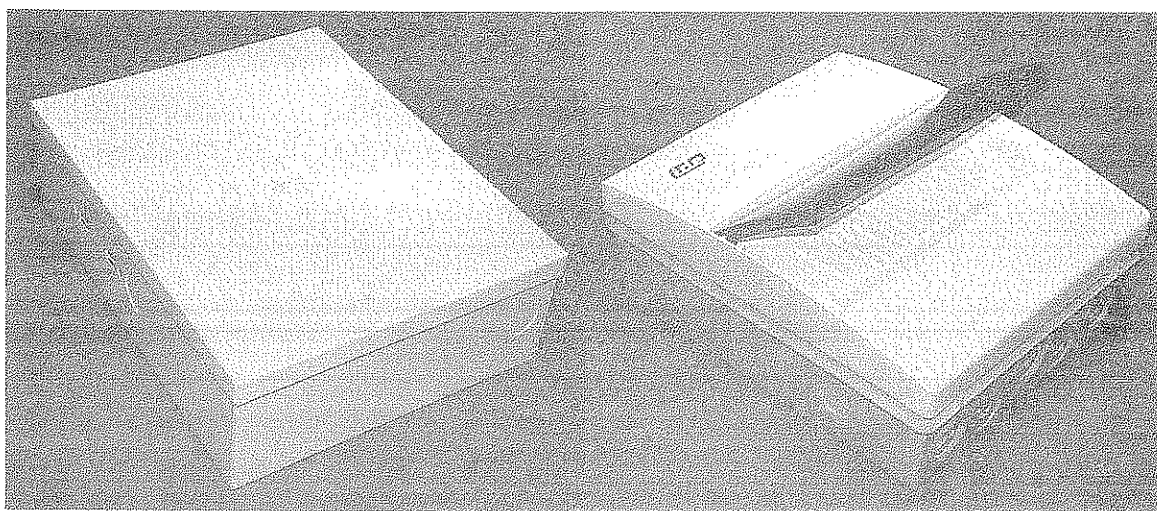


Figura 6.10.5. Fotografía de una caja metálica y otra de plástico, ambas se utilizan para instalaciones eléctricas, con la caja de la izquierda haremos una alarma de 3 zonas y otra de 9 zonas y con la de la derecha haremos un PLC con LCD.

Para hacer la caja de la alarma de 3 zonas primero debemos hacer los paneles, estos dibujamos en el computador y lo mandamos a imprimir en papel adhesivo blanco o transparencias.

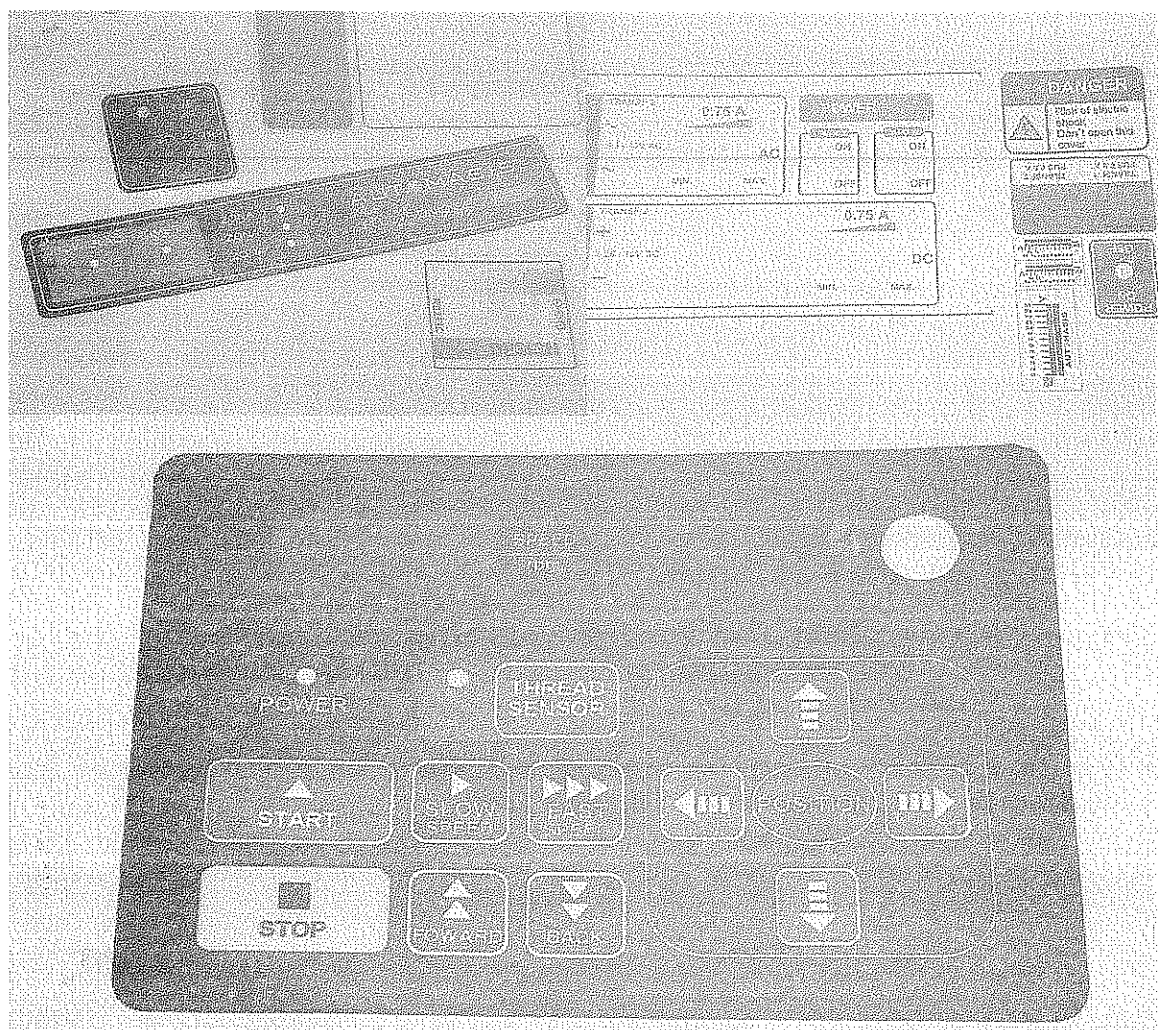
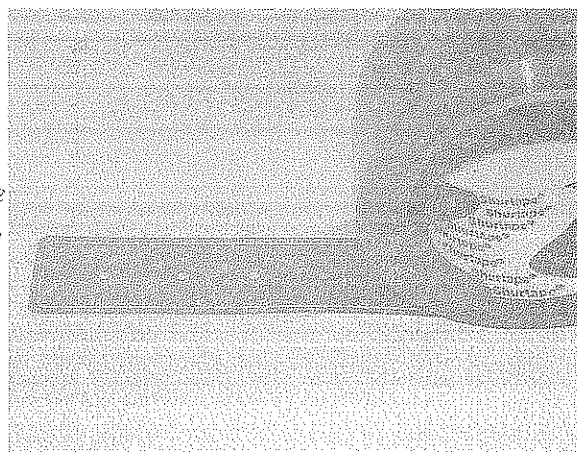


Figura 6.10.6. Fotografía de algunos paneles, unos impresos en papel adhesivo y otro en acetato.

Para proteger los dibujos se debe cubrir con una lámina adhesiva transparente, esto ayudará a que con el tiempo no se borren ni se mojen, ni tampoco se ensucien.



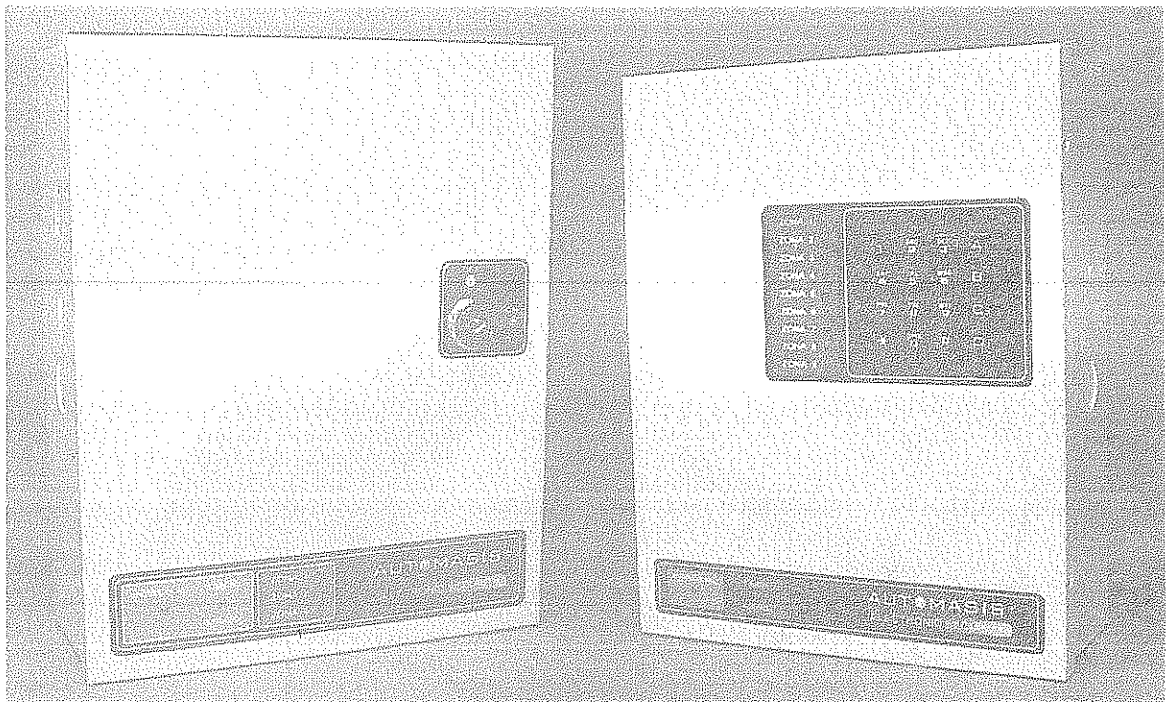


Figura 6.10.8. Fotografía de paneles de una alarma de 3 zonas y otra de 9 zonas con teclado.



Figura 6.10.9. Fotografía de un PLC con LCD fabricado en una caja de breakers de plástico.

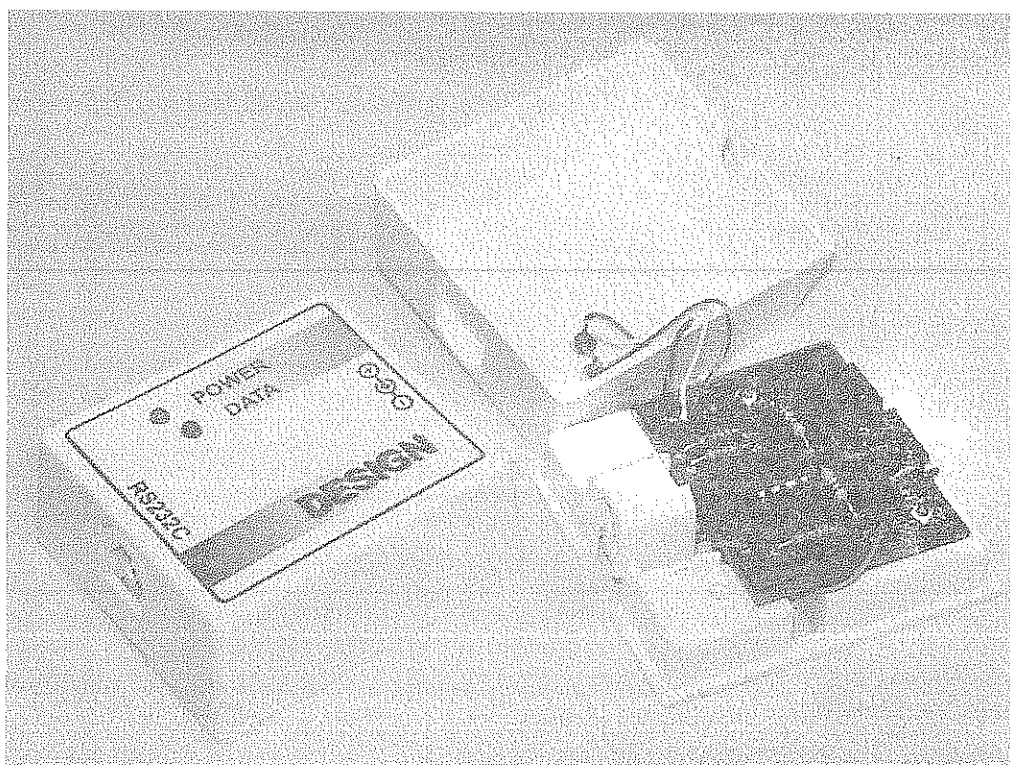


Figura 6.10.10. Un cajetín telefónico, también nos podría servir para contener un proyecto.

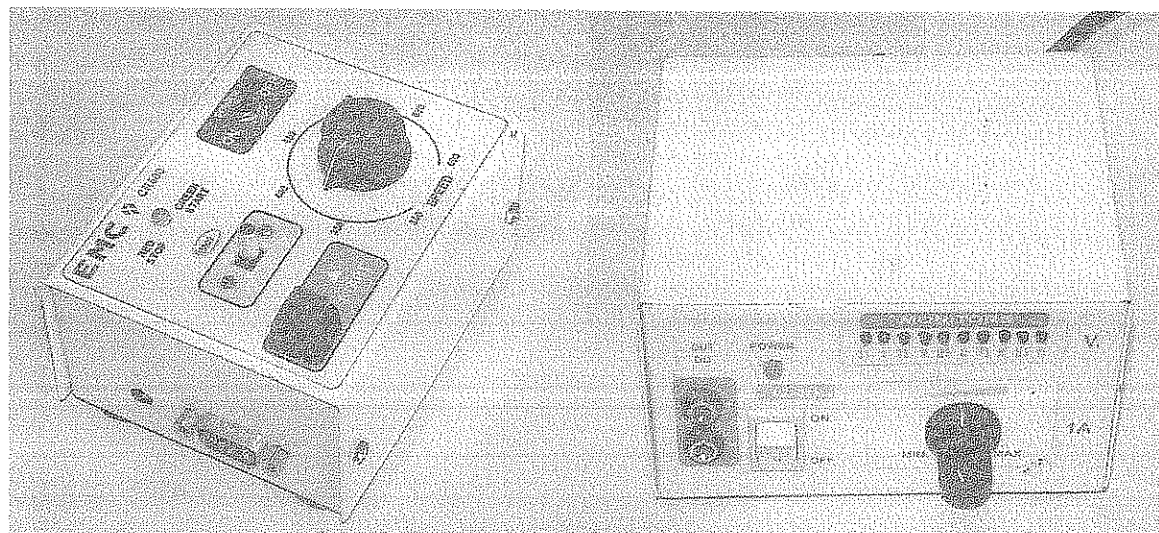


Figura 6.10.11. Las cajas para proyectos que se venden en las tiendas electrónicas, también son una buena opción, algunas vienen con los agujeros listos para colocar switches y leds, y con la ayuda de las impresiones láser a color en papel adhesivo, se les puede dar un acabado muy profesional.

Si la caja que usted necesita no lo encuentra en el mercado, puede mandar a fabricar una a su gusto, por ejemplo para una fuente de poder se necesitaba una caja que tenga un panel de control inclinado, pues se manda a hacer unas tablas con un canal para sujetar los tornillos, y

luego se manda a doblar el tool a la medida correcta, se procede con los agujeros, la pintura y el papel adhesivo, el resultado final puede ser como la siguiente fotografía.

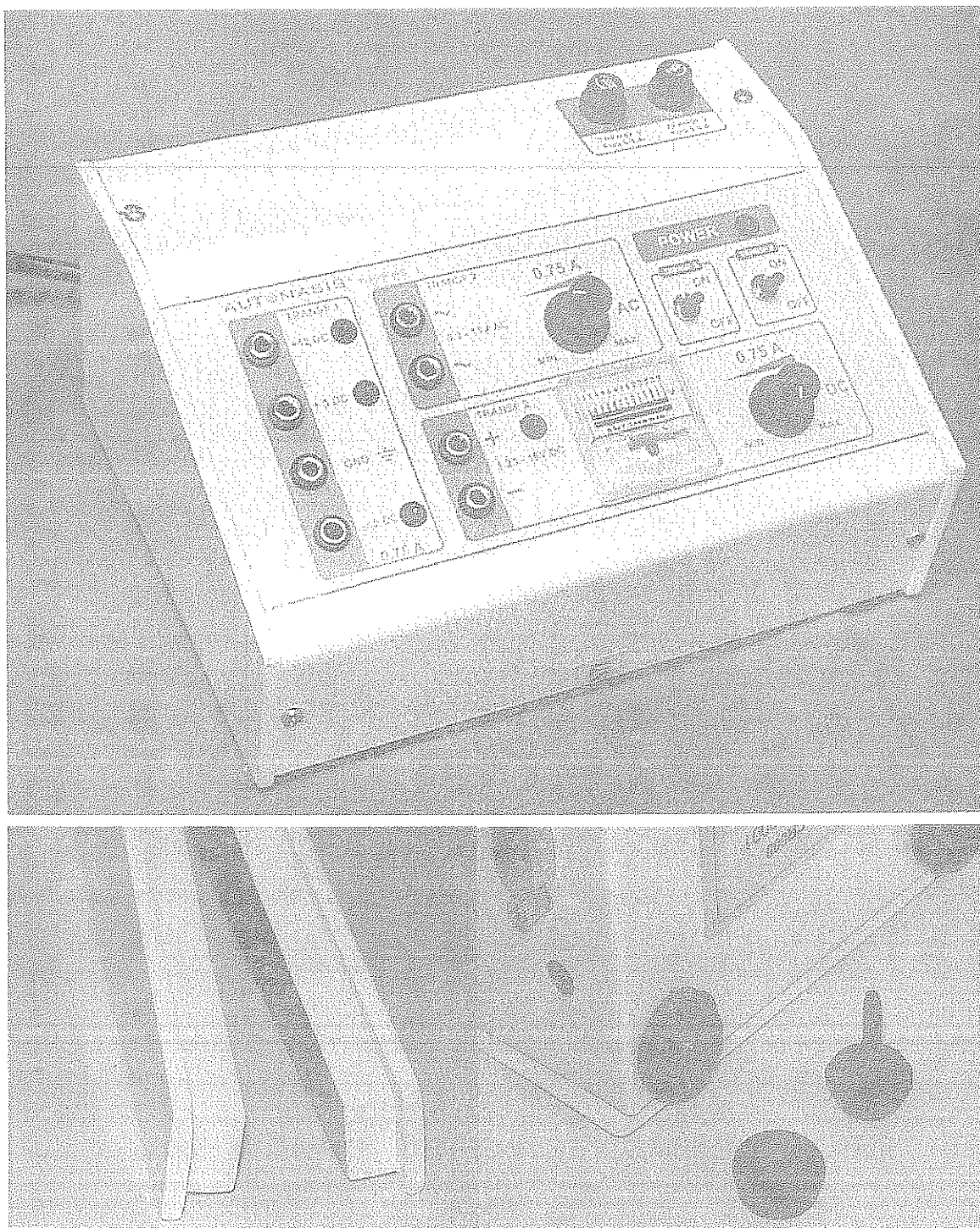


Figura 6.10.12. Chasis de una fuente de poder construida con madera y tool doblado, para las patitas se puede utilizar los cauchos automotrices que se utilizan como topes para las puertas.

*No olvide que usted puede ver todas las fotografías de este subcapítulo en CD\chasis\.

Apéndice A SITIOS WEB RELACIONADOS CON ESTE LIBRO

WWW.mecanique.co.uk	Descarga de programas: Microcode y Pbp demo
WWW.IC-prog.com	Descarga de programa Ic-prog105C.zip
WWW.talquingelectronics.com	Descarga del programa Ic-prog105c-a.zip
WWW.sonsivri.com	Descarga del compilador pbp 2.44
WWW.microchip.com	Productos e información de la familia de PIC'S
WWW.melabs.com	Compilador PicBasic Pro. Ejemplos en pbp
WWW.rentron.com	Ejemplos en pbp, información de dispositivos
WWW.frino.com.ar	Descarga manual de pbp, teoría del PIC
WWW.todopic.com.ar	Descarga manual en español de pbp
WWW.Redeya.com	Historia del PIC
WWW.monografias.com	Todo tipo de temas relacionado al PIC
WWW.x-robotics.com	Información sobre LCD 2x16, motores PAP, etc.
WWW.cipres.cec.uchile.cl	Teoría del bus I2C
WWW.pablin.com.ar	Diagramas de proyectos con PIC'S
WWW.electronicaestudio.com	Productos con y para PIC'S
WWW.mikroelektronika.co.yu	Entrenadores y libros de PIC'S
carlosmicroPIC@hotmail.com	Correo electrónico del autor de este libro

Apéndice B PRÓXIMA ENTREGA

En una próxima entrega se incluirán proyectos con las demás declaraciones del compilador pbp 2.44, y además:

- comunicación serial con LABview
- alarma programable con teclado
- sensores de proximidad
- circuitos grabadores y reproductores de voz
- transmisión y recepción infrarroja
- interesantes proyectos con motores paso a paso
- y muchos proyectos más.