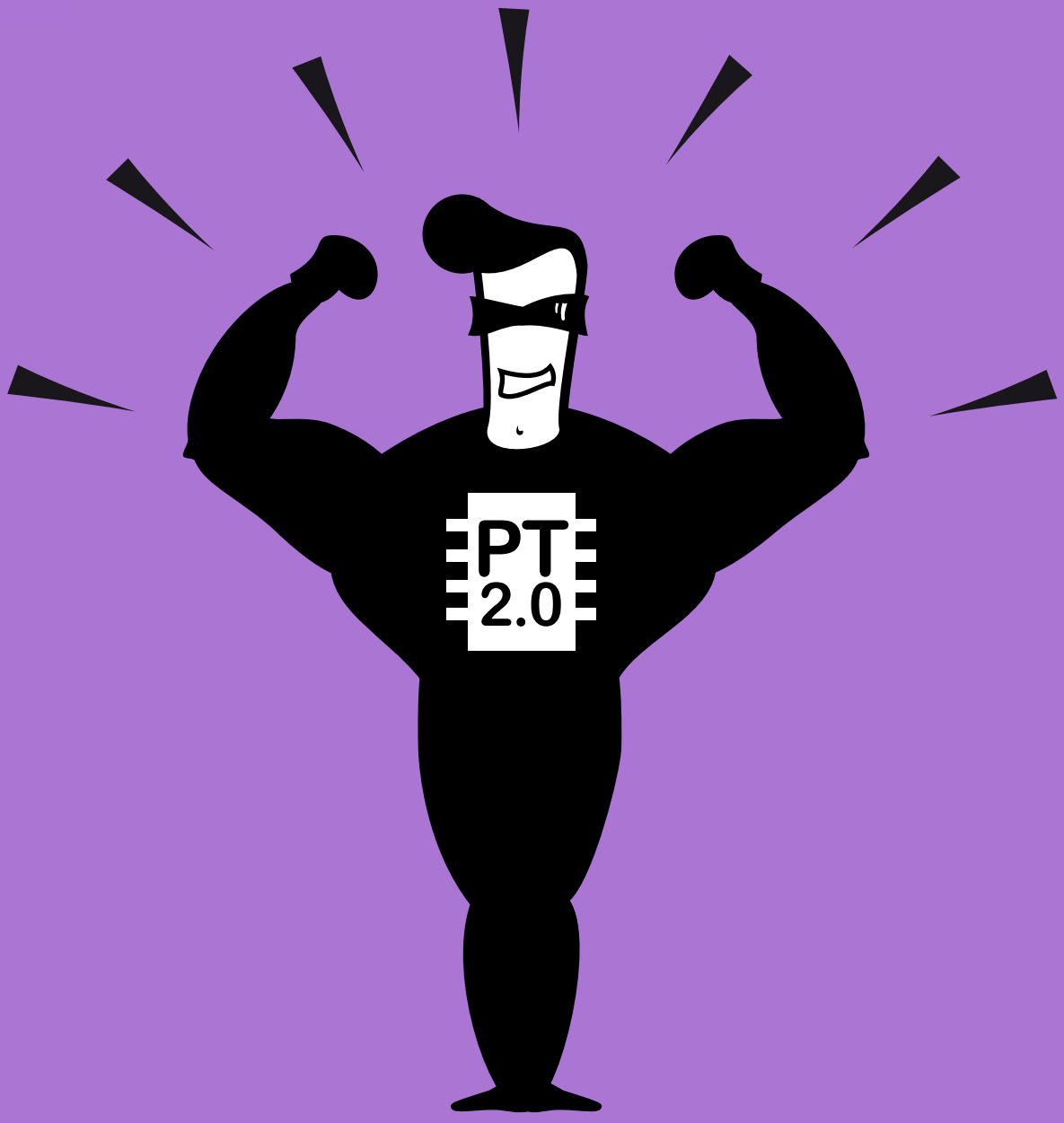
// DESCARGA GRATUITA //

"int año = 1, número 4 " /* JULIO 2008/*

µCONTROL

Electrónica en General Pics en Particular



PIC TRAINER 2.0

Construye tu propia placa entrenadora para microcontroladores PIC.

PIC-Tengu

Un juguete basado en un PIC18F2455

Driver PAP con 74HC194

Cómo construir un sencillo driver para motores paso a paso

.staff

número = 4; año = 1;

Dirección, Redacción y Corrección:
Ariel Palazzesi
Argentina
arielpalazzesi@gmail.com
www.ucontrol.com.ar

Diseño:
DCV Verónica C. Lavore
Argentina
azimut.estudio@gmail.com

Consejo Editorial:
Mario Sacco
Argentina
service.servisystem@gmail.com

Carlos Ortega Sabio
España
carlos.ortegasabio@ucontrol.revista.com.ar

Diego Márquez García - Cuervo

Marcos Lazcano
Argentina
marcos.lazcano@gmail.com

Pedro
Venezuela
palitroquez@gmail.com

.indice

.editorial

Me gusta pensar que si estás leyendo esta editorial, es por que ya has leído antes el resto de la revista. Quizás se deba a que al menos yo leo las revistas de esa manera: comienzo por las partes más jugosas, e indefectiblemente me quedan para el final estas secciones, que muchas veces no se sabe bien para que están.

Bien, hoy quiero aprovechar este espacio para confiarles dos o tres "secretos" relacionados con uControl en general, y con esta revista en particular.

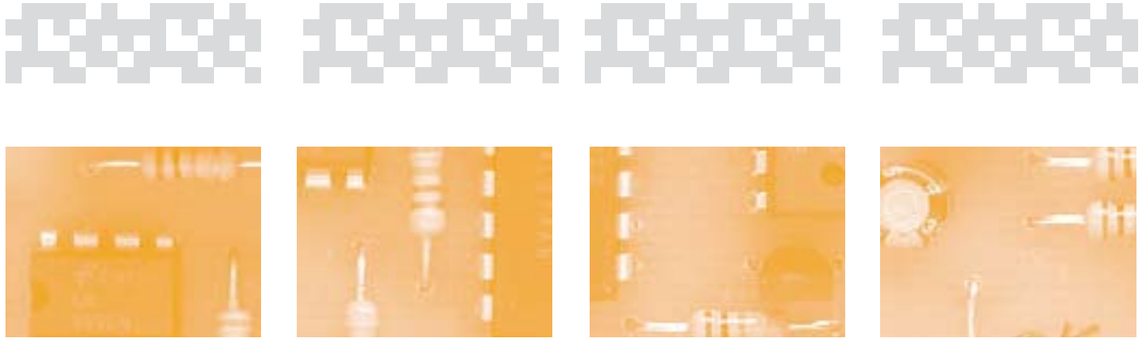
En primer lugar, tengo que confesar que me abruman la cantidad de correos que recibimos cada día preguntando cosas, sugiriendo notas, ofreciendo artículos, y sobre todo, conteniendo palabras de aliento que nos obligan a seguir poniendo lo mejor que tenemos cada bimestre en estas paginas. Sin duda, constituyen la fuerza que nos impulsa a seguir haciendo esto.

En segundo lugar, nos ha llamado la atención la poca cantidad de proyectos que han enviado para el concurso "Cuidemos el planeta". Es posible que no hayamos sabido comunicar correctamente la mecánica del mismo, o que, simplemente, el interés de nuestros lectores por la ecología sea menor del que pensábamos.

En todo caso, les recordamos que todavía hay tiempo para participar, que las bases y condiciones están esperándolos en www.ucontrol.com.ar y, sobre todo, que seria una lástima que por vergüenza o por no realizar un pequeño proyecto se pierdan de ganar alguno de los interesantes premios que tenemos para ustedes.

La tercera cuestión que me gustaría mencionar se relaciona con las colaboraciones que nos envían para su publicación. Muchos se sentirán defraudados al no ver en este número el material (excelente) que nos hicieron llegar. Les pedimos disculpas. El tiempo, como siempre, es escaso, y en muchos casos el poner a punto un proyecto para que pueda ser publicado requiere de tareas de corrección, redibujado de esquemas, trazado de PCB y muchas cuestiones más que hacen imposible tenerlos a tiempo.

driver PAP con 74HC194

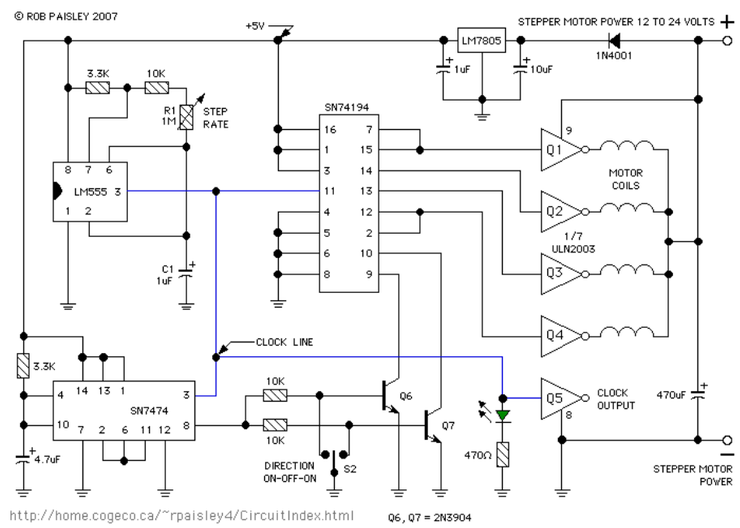


En este artículo veremos como construir un sencillo y razonablemente económico driver para motores paso a paso del tipo unipolar. Dicho driver podrá utilizarse con motores de baja potencia y no necesita de un microcontrolador para funcionar. Puede ser un buen punto de partida para diseñar tu propio driver.

//por:Rob Paisley//

El circuito se basa en el circuito integrado 74LS194 (un registro de desplazamiento bidireccional)

|| FIGURA 1 || ***Este es el circuito de nuestro driver.***



El circuito se basa en circuito integrado 74LS194 (un registro de desplazamiento bidireccional). Está diseñado para ofrecer las funciones básicas de control, como Avance (Forward), Retroceso (Reverse), Parada (Stop) y ajuste de la velocidad de giro en hasta 100 pasos por minuto.

No se trata de un proyecto complejo, y todas las partes empleadas pueden conseguirse con facilidad. El método elegido para alterar el sentido de giro es una llave, pero como verás, es muy fácil cambiar esto para que pueda ser controlado desde el puerto paralelo de un ordenador o desde un microcontrolador.

El control de la velocidad se realiza mediante un potenciómetro, aunque también podría emplearse para ello un dispositivo como los mencionados anteriormente.

.Circuito básico del driver

La figura siguiente nos muestra el circuito básico empleado. En color azul puede verse la línea de CLOCK.

Un circuito integrado NE555 funcionando como oscilador estable proporciona los pulsos de CLOCK necesarios y se envían al pin 11 del circuito integrado 74LS194.

Cada vez que CLOCK esta en alto (positivo) el estado de las salidas del 74LS194 (pines 12, 13, 14 y 15) son rotadas. Puedes consultar el diagrama que aparece más abajo para ver los detalles.

La dirección de esta rotación se determina mediante la llave S2. Cuando S2 está en la posición central (OFF), el motor se detiene.

Cuando la base del transistor Q6 esta a nivel bajo, las salidas del 74LS194 cambian en el orden 12 - 15 - 14 - 13 - 12 , etc.

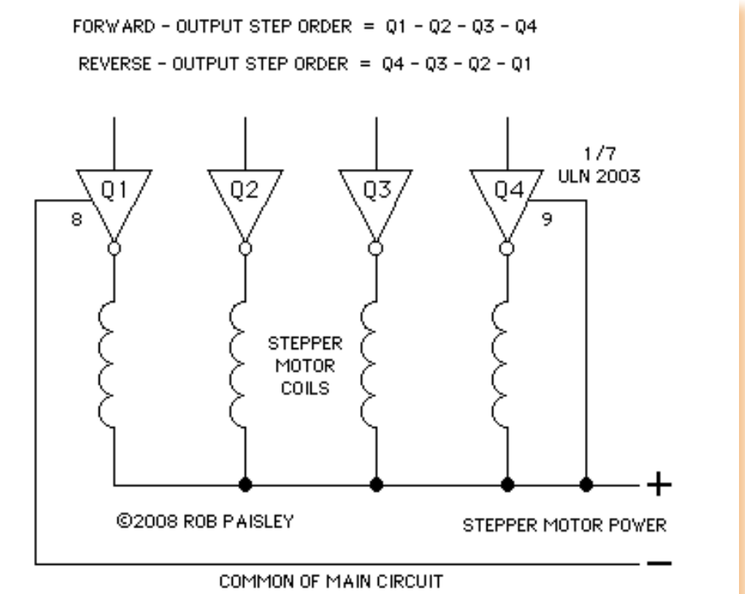
Cuando la base del transistor Q6 esta a nivel alto, las salidas del 74LS194 cambian en el orden 12 - 13 - 14 - 15 - 12 , etc.

Los pulsos existentes en las salidas del 74HC194 se envían al motor a través de un circuito integrado ULN2003. Este se encarga de manejar la corriente necesaria para excitar las bobinas.

.Circuitos integrados empleados

- 74LS194, registro de desplazamiento bidireccional de 4 bits
- 74LS74, Doble flip-flop tipo D con Preset y Clear
- ULN2003, Driver darlington de 7 canales, 500mA por canal, 50V máximo.
- NE555, configurado como oscilador astable.

|| FIGURA 2 || ***Este es el circuito de nuestro driver***



El diagrama siguiente muestra la forma en que deben energizarse las salidas del ULN2003 para hacer girar el motor hacia delante y hacia atrás. Los números de los pines no se han incluido ya que el conexionado final dependerá del diseño del PCB.

Cada pulso positivo en las salidas del 74LS194 provocará el encendido de una de las bobinas del motor.



.Notas sobre el circuito

A continuación, hemos reunido una serie de datos importantes sobre el circuito propuesto y su funcionamiento:

- *Con los valores indicados en el esquema y C1 = 1 uF, un valor de R1 = 0 ohm hará que la frecuencia de CLOCK sea aproximadamente de 100KHz. Esto hace que el motor avance unos 100 pasos por segundo, velocidad limite para la mayoría de los motores paso a paso.*

El 74LS74 no cumple otra función más que proporcionar el control del 74LS194, fijando la dirección de giro.

- *Si se incrementase la velocidad, no solo disminuiría el torque disponible sino que también se correría el riesgo de que el motor "pierda" pasos. Se pueden probar diferentes valores para C1 y R1 para producir la frecuencia máxima mas adecuada para cada motor en particular. Este valor dependerá exclusivamente de las características constructivas del mismo.*

- *Si R1 adopta un valor cercano a 1 Megohm, la frecuencia del CLOCK disminuirá a cerca de un Hz, haciendo que el motor avance un paso por segundo.*

- *No hay, al menos en teoría, un valor mínimo de velocidad a la que pueda girar un motor paso a paso. Esto permite utilizar valores de C1 y R1 tan grandes como se desee, siempre dentro de los valores aconsejados en la hoja de datos del NE555.*

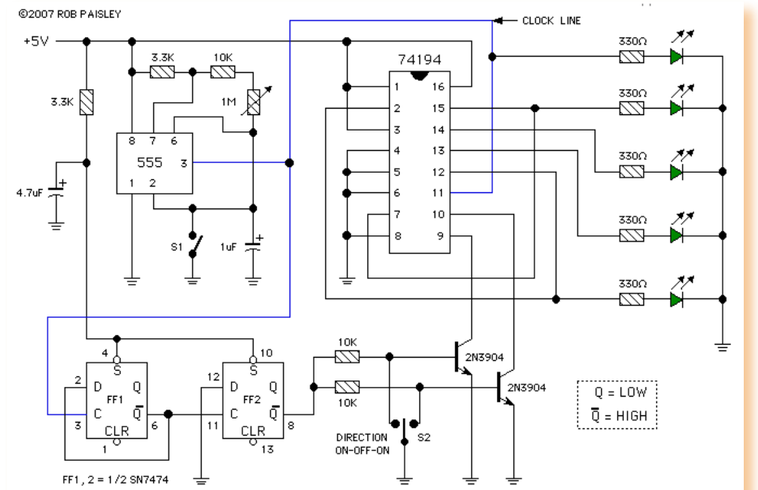
- *El circuito impreso propuesto permite la conexión de la resistencia variable R1 en una bornera, que también puede emplearse como punto de entrada para pulsos de control externos.*

- *La llave S1, que aparece en el diagrama de más abajo, permite la detención del motor al interrumpir la generación de pulsos de CLOCK del NE555. S1 puede ser reemplazado por un transistor NPN para controlar electrónicamente la generación de pulsos de CLOCK.*

- *Los pulsos de CLOCK pueden ser provistos por un circuito externo, pero cualquier ruido en estos podría colocar al registro de desplazamiento en un valor erróneo. En caso de emplearse, deben ser pulsos "limpios". Estos pulsos deberían pasar a través del NE555, situación que esta prevista en el circuito impreso.*

- *El 74LS74 no cumple otra función más que proporcionar el control del 74LS194, fijando la dirección de giro con la ayuda de los transistores Q6 y Q7 y la llave S2. Estrictamente hablando, el método de control que ofrece te sistema no es el mejor, pero a las relativamente bajas frecuencias a las que opera el circuito (menores a 100KHz) funciona perfectamente.*

II FIGURA 3 II ***De esta forma podemos probar nuestro driver***



- *El resistor de 3.3K y el condensador de 4.7 uF conectados en el terminal SET (pines 4 y 10 del 74LS74) aseguran que las salidas (pines 6 y 8) permanezcan en estado bajo durante el encendido del circuito.*

- *Al alimentar el circuito, es posible que no todas las salidas del 74LS194 estén en estado bajo. Por este motivo, la fuente de poder que alimente el circuito debe ser capaz de soportar la carga de las cuatro bobinas energizadas al mismo tiempo, durante un instante más o menos largo (dependiendo de la posición de R1).*

- *Este circuito puede ser comandado desde otro, o desde el puerto paralelo de un ordenador. Para que este sistema de control externo funcione, se debe garantizar que la base de los transistores Q6 y Q7 sea de al menos 0.7V. Puede que sea necesario utilizar un transistor adicional para lograr este objetivo.*

- *En caso de energizar simultáneamente los transistores Q6 y Q7, el 74LS194 efectuara un RESET, deteniendo el giro del motor y energizando su salida numero 15 cuando reciba el próximo pulso de CLOCK.*

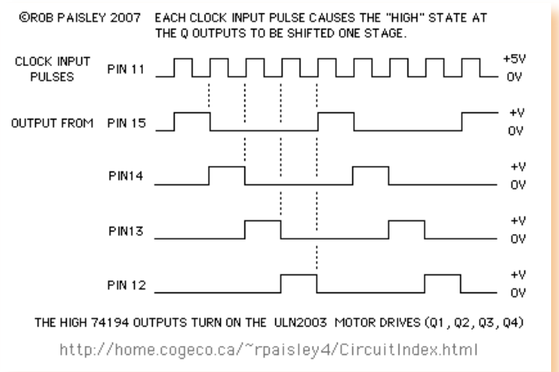
- *El circuito necesita de una fuente de 5V de corriente continua perfectamente estabilizada para funcionar, que no se ha incluido en el esquema.*

- *Existen una gran variedad de motores paso a paso. Debe asegurarse que el elegido para ser empleado con este sistema se alimente con tensiones inferiores a los 24V y no consuma mas de 500mA.*

.Probando el controlador

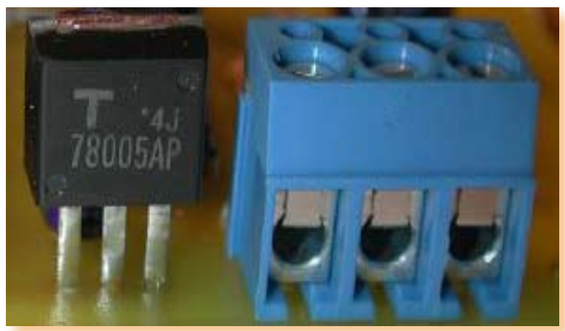
Simplemente colocando un LED en cada salida podemos comprobar visualmente el funcionamiento del circuito.

En el esquema pueden verse los dos Flip Flop "D" incluidos en el 74LS74. La sección FF1 se utiliza como un divisor binario, mientras que FF2 funciona como un Flip Flop "RS". Después de cada pulso de CLOCK, el Flip Flop es puesto



II FIGURA 4 II ***formas de onda que se pueden encontrar a la salida del driver***

II FIGURA 5 II ***Aspecto del driver ya montado***



II FIGURA 6 II ***Para reducir la altura del circuito final, puede cortarse la aleta disipadora del LM7805***

El 74LS74 no cumple otra función más que proporcionar el control del 74LS194, fijando la dirección de giro.

en SET, con Q en alto. Esto permite comandar al 74LS194 secuenciar sus salidas en uno u otro sentido de acuerdo a la posición de S2.

La llave S1 permite detener la generación de pulsos de CLOCK.

Los terminales POWER (14), COMMON (7) y CLEAR (1 y 13) del 74LS74 no se muestran, pero los correspondientes a CLEAR deben conectarse a +5V.

El siguiente grafico muestra las formas de onda que se pueden encontrar a la salida del driver: (Ver Figura 4).

Este es el aspecto del driver ya montado. (Ver Figura 5 y 6)

.Alejando el motor del driver

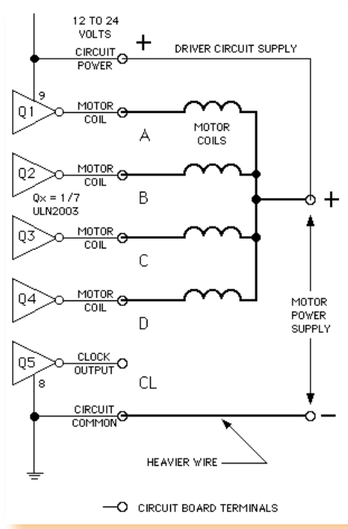
Si el motor va a emplearse a una distancia importante del controlador, deben separarse los bornes de alimentación, tal como se ve en el esquema: (Ver Figura 7)

Esto evita que los pulsos generados por el motor introduzcan ruidos en la fuente de alimentación.

.Motores de 6 terminales

Algunos motores paso a paso disponen de 6 terminales. La siguiente imagen muestra la forma en que deberían conectarse al controlador: (Ver Figura 8)

II FIGURA 7 II ***Así podremos alejar el PAP del driver***

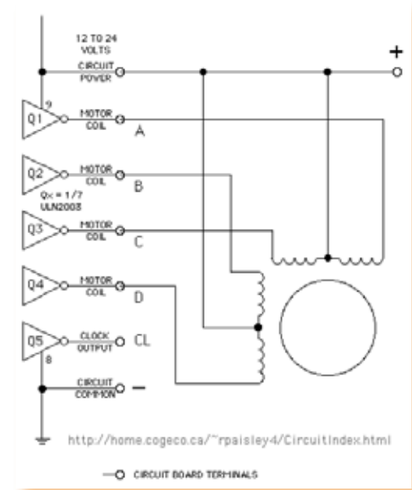


.Control externo utilizando transistores

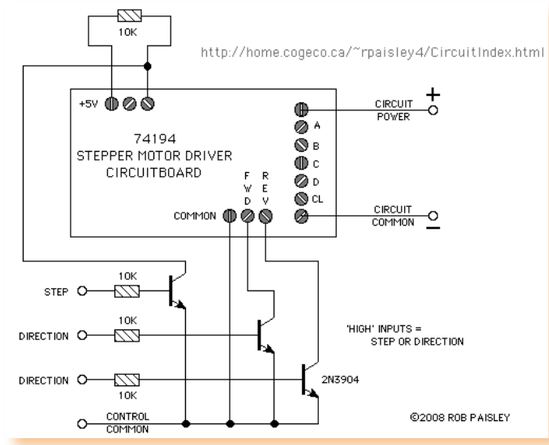
Como se mencionó antes, es posible comandar el driver mediante la utilización de transistores. Este sistema permite operar la placa controladora desde otro circuito o desde un ordenador:
(Ver Figura 9)

.Control externo mediante optoacopladores

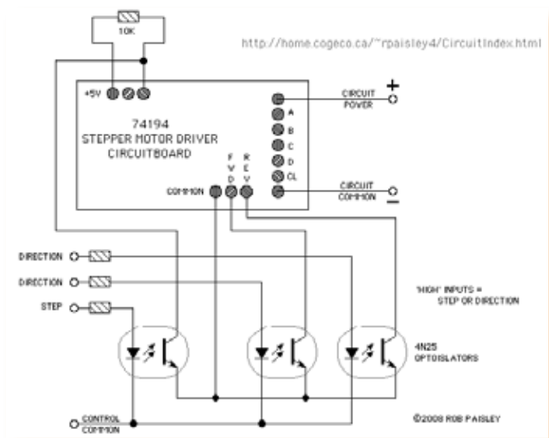
Otra forma de control, utilizando optoacopladores para aislar el driver del circuito que lo controla.
(Ver Figura 10)



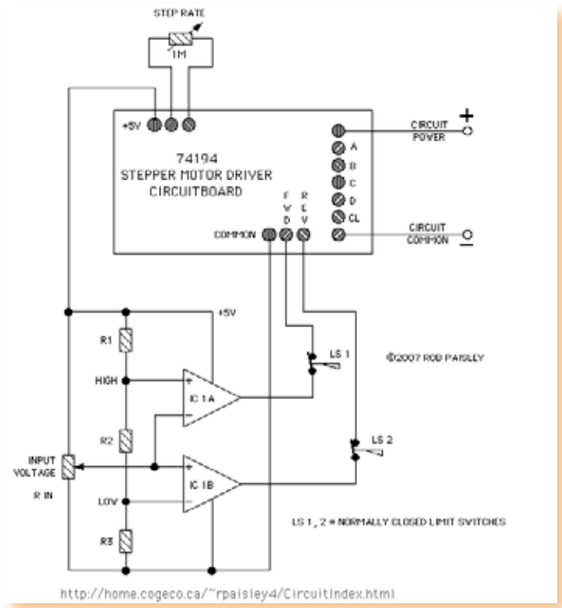
II FIGURA 8 II ***Los cables centrales son comunes a dos bobinas***



II FIGURA 9 II ***Puedes comandar el driver externamente***



II FIGURA 10 II ***Los optoacopladores evitan posibles daños al host.***



II FIGURA 11 II ***Es posible automatizar su funcionamiento***

Lista de componentes:

1	x	74LS194
1	x	74LS74
1	x	ULN2003AN
1	x	NE555N
1	x	L7805ACV
2	x	2N3904
1	x	512-1N4001
1	x	470uF/35V
1	x	10uF/25V
1	x	4.7uF/25V
1	x	1uF/25V
1	x	GREEN 3mm LED
3	x	10K 1/4W
2	x	3.3K 1/4W
1	x	470 OHM 1/4W
2	x	2 POS. TERMINAL BLOCK
3	x	3 POS. TERMINAL BLOCK

.Control automático

Este circuito reemplaza la llave S2 por un control automático basado en dos amplificadores operacionales. Esto brinda una "ventana" dentro de la cual el motor girará en uno u otro sentido.

El potenciómetro R IN puede ser reemplazado por un sensor de temperatura o de luz. LS1 y LS2 funcionan como sensores de fin de carrera, que evitan que el motor continúe girando más allá del punto fijado. (Ver Figura 11)

Cada pulso positivo en las salidas del 74LS194 provocará el encendido de una de las bobinas del motor.

.Información adicional

■ Página web del autor (Ingles): http://home.cogeco.ca/~rpaisley4/CircuitIndex.html

■ El driver ha sido probado con los siguientes motores:

JAPAN SERVO CO. (de un viejo floppy drive)
TYPE KP4M4-001
75 OHM / PHASE
0.15 AMP / PHASE

y

AIRPAX : LA82720-M1
24 VOLT
60 OHMS / COIL
7.5 DEGREES / STEP

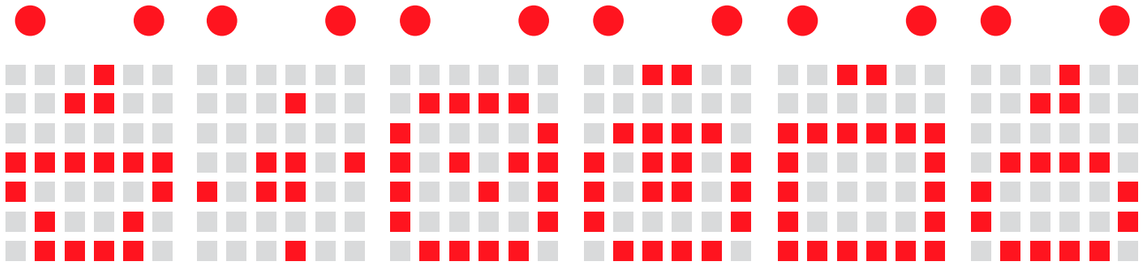
En esta ocasión, y debido a que el autor del artículo vende en su página web el PCB para montar el proyecto, no podemos proporcionar el diseño del mismo. Pero estamos seguros que los lectores de ***uControl*** no tendrán problemas a la hora de diseñar uno propio.■

NOTA: *Debido a la falta de detección o corrección de errores y la potencia de salida limitada, este circuito no debe ser utilizado para aplicaciones que requieren una gran precisión en el control o posicionamiento. El driver está pensado para el aprendizaje y la utilización en pequeños robots de aficionados.*

//página 0x0A

PIC-Tengu

primera parte



PIC-Tengu es un juguete basado en un PIC18F2455 capaz de mostrar gestos en su matriz de LEDs al ritmo de los sonidos ambientales que recoge su micrófono electret. Además, incluye un sencillo juego que lo convierte en un original regalo de cumpleaños.

//por:Juan Félix Mateos//
jfmateos@lycos.es

El trabajo de Crispin Jones fue la inspiración de este proyecto, enfocado desde un punto de vista eminentemente didáctico.

Este artículo debe comenzar con el merecido reconocimiento al creador del Tengu original (www.tengutengutengu.com): el joven diseñador británico Crispin Jones (www.mr-jones.org). Su trabajo desborda ingenio, creatividad e innovación, y fue la inspiración para abordar este proyecto desde un punto de vista eminentemente didáctico.

Antes de adentrarnos en cuestiones técnicas sentaremos las bases dando respuesta a tres preguntas esenciales: qué vamos a hacer, por qué lo vamos a hacer y cómo lo vamos a hacer. En otras palabras, describiremos el funcionamiento de PIC-Tengu, diseñaremos una estrategia para abordar su desarrollo y elegiremos los medios adecuados para conducir esta nave a buen puerto.

.Descripción de PIC-Tengu

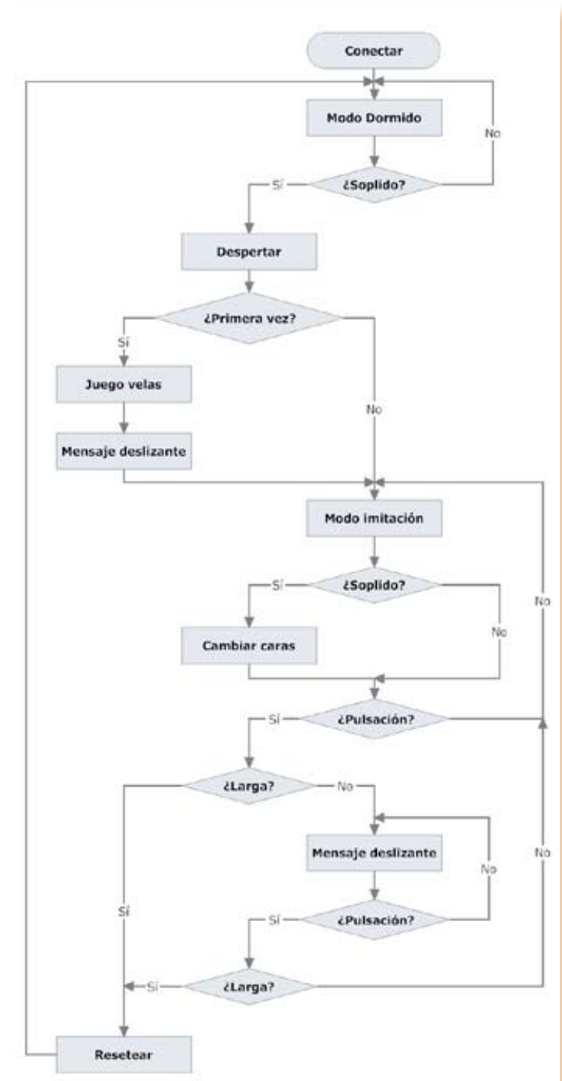
La misión esencial de PIC-Tengu será hacer gesticular las caras que muestra en su matriz de LEDs al ritmo de los sonidos que escuche a través de un micrófono. No obstante, lo dotaremos de otras funcionalidades secundarias que, no por ser de menor importancia, debemos dejar de lado en una fase tan incipiente del proceso de diseño. Ignorar estas funciones menores podría hacer avanzar nuestro proyecto en una línea que posteriormente imposibilitase su implementación.

Además de su micrófono electret y de su matriz de LEDs, PIC-Tengu contará con 2 LEDs adicionales que simularán sus ojos, y con un botón que ofrecerá una vía alternativa de interacción con el usuario.

El funcionamiento completo de PIC-Tengu será el siguiente:

1. Al conectarlo a la fuente de alimentación PIC-Tengu se mostrará en estado dormido (con los ojos cerrados/apagados y con la boca cerrada).
2. Para despertarlo tendremos que soplarle y detectaremos esta situación aprovechando la saturación que producirá en el micrófono.
3. Entonces PIC-Tengu iniciará su secuencia de despertar, que consistirá en una sencilla animación que simulará cómo los ojos se abren poco a poco, seguidos de un bostezo y varios pestañeos consecutivos.
4. Si es la primera vez que se enciende, PIC-Tengu iniciará un sencillo juego que consistirá en ir mostrando velas encendidas para que el usuario las apague a soplidos una a una. Al finalizar el juego se mostrará en la matriz de LEDs un mensaje de felicitación deslizante. Este mensaje podrá mostrarse en cualquier otro momento a petición del usuario, como se explicará posteriormente.
5. Al concluir la presentación del mensaje deslizante, o si no es la primera vez que lo encendemos, PIC-Tengu iniciará el modo de imitación, que consistirá en gesticular al ritmo del sonido mediante diversos juegos de caras. El cambio de juego de caras podrá ser solicitado por el usuario soplando, o se producirá automáticamente cuando la intensidad del sonido sea muy alta. Además de las caras, también utilizaremos los ojos para reforzar el efecto de imitación, haciéndolos parpadear tanto más rápido cuando más intenso sea el sonido.
6. Si transcurriesen más de 5 minutos sin que PIC-Tengu detectase sonido alguno, volvería nuevamente a su estado dormido, del que tendríamos que sacarle soplando.
7. Pulsando brevemente el botón, el usuario podrá alternar entre el modo de imitación y el modo de mensaje deslizante.
8. Manteniendo el botón pulsado más de 2 segundos, PIC-Tengu se restablecerá, adoptando el mismo estado que si lo encendiésemos por primera vez.
9. Para afrontar futuras ampliaciones, nos interesa dejar abierta la posibilidad de comunicarnos con PIC-Tengu a través de un ordenador.

|| FIGURA 1 || ***Tengu original a la izquierda y nuestro PIC-Tengu a la derecha.***



|| FIGURA 2 ||
Diagrama de flujo del funcionamiento de PIC-Tengu.

.Finalidad del proyecto

La finalidad de este proyecto es esencialmente didáctica, por lo que se huirá de aplicar técnicas complejas a pesar de que pudieran proporcionar un mejor rendimiento.

El objetivo es poner de manifiesto mediante ejemplos prácticos y sencillos las posibilidades de algunos de los módulos incluidos en los microcontroladores PIC, como:

- ***Salidas digitales:*** PIC-Tengu cuenta con 44 LEDs (7 filas y 6 columnas en la matriz, más los dos ojos) y todos ellos serán controlados mediante salidas digitales del PIC.
- ***Conversor analógico digital:*** Lo utilizaremos para detectar la intensidad del sonido.
- ***Pulse Width Modulation:*** Nos permitirá variar la intensidad de los LEDs que simulan los ojos para aparentar que se abren poco a poco.
- ***Timers:*** Gracias a ellos podremos actualizar la matriz de LEDs a un ritmo constante aunque tengamos que realizar simultáneamente otras tareas. También los utilizaremos para averiguar si el botón ha permanecido pulsado más de 2 segundos.
- ***Interrupción externa:*** Nos permitirá detectar la pulsación del botón en cualquier instante, aunque en ese momento se estén realizando otras tareas.
- ***Memoria EEPROM:*** Utilizaremos esta franja de memoria para almacenar la edad del homenajeado y el mensaje deslizante, aprovechando así sus dos cualidades más destacables: su contenido no se borra aunque cese la alimentación eléctrica, y podemos alterar su contenido en tiempo de ejecución desde el propio firmware del PIC sin tener que reprogramarlo.

.Medios

En primer lugar necesitamos elegir el PIC más apropiado, es decir, aquél que ofrezca todas las características señaladas en la sección anterior por el precio más económico.

No obstante, aún nos queda por analizar con más detalle algunos parámetros, como el número de entradas/salidas digitales que necesitaremos. Nuestra primera tentación podría ser utilizar un pin para controlar cada LED independientemente, pero esto nos conduciría a un sobredimensionamiento del PIC injustificado, pues debemos tener en cuenta que cada pin puede suministrar o absorber un máximo de 25 mA, y que los PIC en general pueden suministrar y absorber un total máximo de 200 mA; considerando que cada LED consume unos 10 mA sólo podríamos tener encendidos un máximo de 20 a la vez, de modo que utilizar un pin para cada LED no nos aportaría ninguna ventaja y supondría una complicación extraordinaria en el ruteado del PCB.

Un detalle a tener en cuenta sobre el ULN2803, en la fase de programación, es que utiliza lógica inversa.

¿Cómo podemos abordar este problema? Multiplexando la matriz, es decir, encendiendo en cada instante sólo los LEDs de una fila o columna, y aprovechándonos de la persistencia de la visión (POV) del ojo humano para ir cambiando de fila o columna activa a suficiente velocidad como para engañar al cerebro y hacerle creer que hay LEDs encendidos a la vez en todas la matriz. Este sistema requiere un pin para cada fila y columna, de modo que necesitaríamos un total de 7 + 6 = 13 pines para controlar toda la matriz.

La siguiente pregunta sería ¿multiplexamos por filas o por columnas? En ambos casos se superaría el límite de corriente que es capaz de absorber un solo pin, de modo que es evidente que vamos a necesitar un driver intermedio entre la matriz y el

Para que la conexión USB sea operativa no debemos olvidar colocar en el pin 14 un condensador de 470nF a tierra.

PIC. Podríamos implementar este driver mediante transistores, pero esto supondría un aumento considerable del tamaño del PCB, pues además de los transistores deberíamos incluir resistencias en sus bases. La solución es recurrir a un circuito integrado que ofrezca sumideros de corriente NPN tipo Darlington. Una vez detectada la necesidad de este driver intermedio, la elección entre filas o columnas se decanta lógicamente por estas últimas, pues al ser menos (6 columnas frente a 7 filas) podremos refrescar la matriz completa en menos tiempo, reduciendo así el riesgo de que se produzcan parpadeos. Idealmente, lo más compacto hubiera sido utilizar un ULN2003, que contiene 7 drivers Darlington en tan sólo 16 pines, pero la dificultad para localizar este integrado nos condujo al ULN2803, que ofrece un octavo driver a costa de incrementar en 2 su número total de pines (hasta 18).

Recordemos que todas estas pesquisas iban encaminadas a dilucidar el número de entradas/salidas que debería tener nuestro PIC: 6 para controlar la columna activa, más 7 para controlar los LEDs de la columna activa, más 2 para los LEDs de los ojos (que deberán actuar en modo PWM), más uno para la entrada del micrófono (que deberá actuar en modo ADC), y más uno para el botón (que deberá ser capaz de detectar una interrupción externa), nos dan un total de 17 pines.

A pesar de toda la deducción anterior, lo cierto es que podría haberse prescindido del driver intermedio, pues el tiempo que estará encendida cada columna será tan corto que producirá un pico de corriente breve que sí podría ser asimilado directamente por un pin del PIC. No obstante, se insiste en que el fin de este proyecto es didáctico.

Otro detalle que también debemos especificar con más detalle es el tipo de comunicación que deseamos establecer con el PC. En los tiempos que corren, la opción casi forzosa es recurrir a una conexión USB, pues cada vez son menos los ordenadores que ofrecen puertos serie RS232. Además, aprovechando esta conexión podremos tomar la alimentación directamente del ordenador, que ya estará adaptada a los 5 V con los que haremos funcionar el PIC.

Recurrimos al buscador paramétrico que Microchip nos ofrece en su web introduciendo los siguientes criterios uno a uno:

- ***Estado: En producción***
- ***Arquitectura: 8 bits***
- ***Compatibilidad USB: USB 2.0***
- ***Memoria de datos EEPROM: 256 bytes***
- ***Entradas digitales: 24***

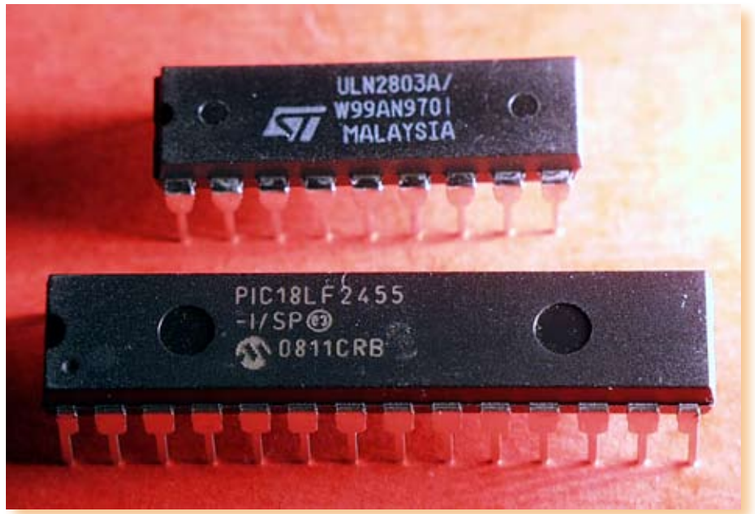
Y dentro de la lista de 4 resultados posibles, como todos ellos ofrecen idénticas prestaciones de timers, ADC y CCP (Capture/Compare/Pulse Width Modulation), que además son suficientes para nuestros propósitos, nos decantamos por el más económico: el PIC18F2455.

Una vez elegido el PIC, debemos decidir cuál será el lenguaje de programación. En este asunto existen opiniones muy diversas, pero lo que nadie pone en cuestión es que los lenguajes de alto nivel facilitan tan notablemente la fase de desarrollo, que el uso de ASM sólo está justificado en aplicaciones que requieran un control muy estricto de tiempos (por ejemplo, para generar señales de vídeo), y éste no es nuestro caso. Los 2 lenguajes de alto nivel más populares para PICs son BASIC y C. Aunque el primero es realmente muy sencillo, lo cierto es que C ofrece una mejor gestión de las interrupciones, y por eso será nuestro elegido, y más concretamente CCS C.

Respecto a los otros 2 elementos necesarios para desarrollar el proyecto, a saber, el software de diseño y el programador, la decisión es sobre todo una cuestión de gustos y recursos. En este caso se han elegido Proteus para el diseño, y el programador de puerto paralelo GTP Lite en combinación con el software gratuito

Buscando hacer funcionar el PIC a su máxima velocidad (48MHz) colocaremos un cristal de 20MHz.

|| FIGURA 3 || ***ULN2803 y PIC 18F2455.***



Habitualmente los electret cuentan con 2 terminales: tierra, y otro compartido por la alimentación y la señal de salida.

WinPIC800 para la programación.

.Captar el sonido ambiente

El micrófono de tipo electret es la elección ideal para este proyecto por su reducido tamaño, su escaso consumo y su bajo precio. Estos micrófonos están basados en los clásicos micrófonos de condensador; de hecho, su nombre completo es ECM (Electret Condenser Microphone). Sin embargo, se diferencian de ellos en que incluyen un material capaz de almacenar una carga eléctrica durante cientos de años; este material se denomina electret y provee la tensión necesaria para polarizar el condensador, de modo que no requerirían alimentación externa. Sin embargo, la tensión es tan baja que se podría ver afectada por las etapas posteriores, de modo que se utiliza un transistor de tipo JFET a modo de adaptador de impedancias, y este transistor sí requiere alimentación externa.

Habitualmente los micrófonos electret cuentan con 2 terminales, uno para la conexión a tierra y otro compartido por la alimentación y la señal de salida (aunque también los hay con 3 terminales en los que la alimentación y la señal están independizadas). Para identificar los terminales podemos apoyarnos en el hecho de que el terminal unido a la carcasa del micrófono es la conexión a tierra. La unión entre el terminal y la carcasa suele ser visible, como en la figura 3.

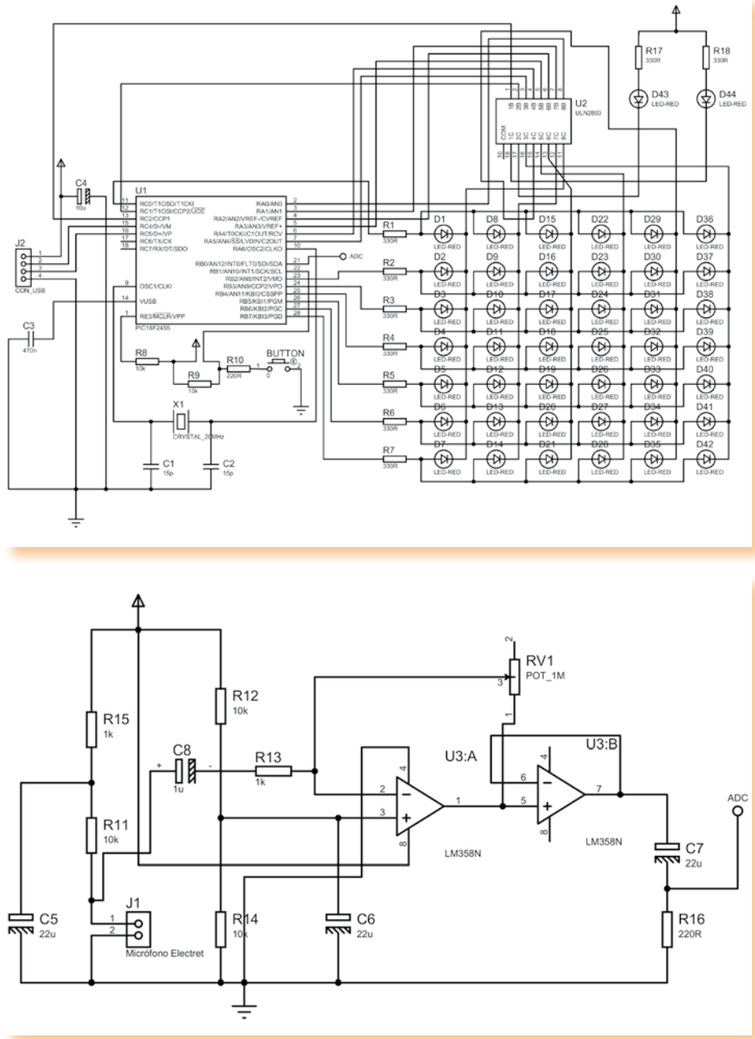
El consumo de los micrófonos electret es del orden de los 0,5 mA, por lo que necesitaremos una resistencia de 10 K en serie con la cápsula para limitar la corriente de la fuente de 5V. No obstante, en serie con esta resistencia colocaremos otra de sólo 1K y, en la unión entre ambas resistencias, un condensador de 22uF con su terminal negativo conectado a tierra. Esta segunda resistencia en combinación con el condensador constituye un filtro básico para evitar que las posibles perturbaciones de la fuente de alimentación pudieran alcanzar al micrófono.

Esta precaución se justifica en el hecho de que el micrófono nos proporcionará una señal de unos pocos milivoltios, de modo que deberemos amplificarla antes de introducirla al PIC. Consecuentemente, cualquier perturbación en la entrada resultaría también amplificada, produciendo una notable degradación de la información de intensidad sonora recogida por el micrófono.

Para amplificar la señal del micrófono recurriremos a un amplificador operacional u op-amp. Tradicionalmente, muchos de los amplificadores operacionales (como el 741) funcionan a partir de fuentes de alimentación de doble raíl, es decir, con una tensión positiva, una tensión negativa y una masa intermedia. Estos circuitos no serían utilizables en nuestro caso, pues disponemos únicamente de la tensión +5V y la tierra que nos proporciona la conexión USB. Consecuentemente, necesitamos un op-amp que funcione a partir de una fuente de un único raíl, con tensiones relativamente bajas, económico, y fácil de conseguir. La respuesta casi natural es el LM358, que contiene en su interior 2 amplificadores. Aunque no se trata de un

II FIGURA 4 II ***Es fácil identificar el terminal de tierra porque está unido a la carcasa del micrófono.***

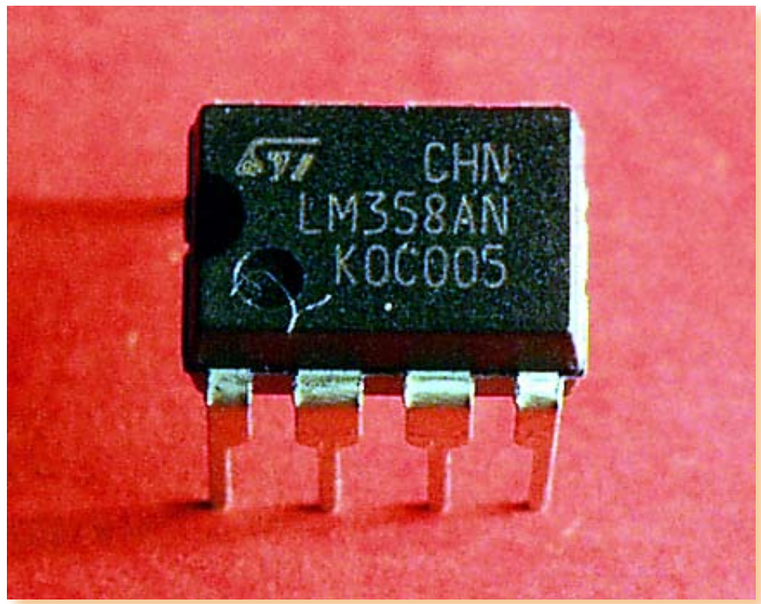
II FIGURA 5 II ***Esquema electrónico de PIC-Tengu.***



amplificador específico para aplicaciones de audio, su estabilidad con ganancias altas lo convierte en una apuesta segura.

Utilizaremos uno de los amplificadores del LM358 como amplificador inversor con realimentación negativa para conseguir amplificaciones controladas, frente a la amplificación de histéresis o saturada (todo o nada) que nos proporcionaría una realimentación positiva.

La realimentación negativa permite domar la fuerte ganancia del amplificador, haciéndolo trabajar en un modo lineal. Al utilizar realimentación negativa, la ganancia se calcula como el cociente entre la resistencia que une la salida del amplificador (pin 1) con la entrada no inversora (pin 2), denominada resistencia de realimentación, y la resistencia a través de la que llega la señal a la entrada no inversora (pin 2), denominada resistencia de entrada. Más exactamente, la ganancia es el opuesto de este número, pues recordemos que estamos haciendo funcionar el amplificador en modo inversor. Como necesitaremos ganancias del orden de 1000 para elevar los milivoltios proporcionados por el micrófono al orden de voltios, el cociente entre estas dos resistencias debe ser 1000, pero ¿qué valores elegir exactamente? Conviene que la resistencia de entrada sea alta para consumir poca corriente de la fuente (el micrófono), pero si la resistencia de realimentación es muy alta pueden producirse inestabilidades. Consecuentemente, una solución de compromiso puede ser elegir una resistencia de entrada de 1K; esta elección es muy frecuente pues la impedancia de salida de muchos equipos como reproductores de CD es del orden de este valor. De este modo, la resistencia de realimentación debería ser un potenciómetro de 1M. Pese a que la ganancia teórica puede ser muy alta, la señal amplificada nunca superará los niveles impuestos por la tensión de alimentación del amplificador.



|| FIGURA 6 ||
El LM358 es un amplificador que funciona con tensiones bajas y proporciona ganancias altas. Existen otros ICs específicos para aplicaciones de audio, como el LM386, pero que están más pensados para alimentar etapas de potencia que para preamplificar señales.

Antes de introducir la señal del micrófono al amplificador la obligaremos a atravesar un condensador de 1uF, de modo que quede despojada de cualquier nivel de continua que pudiera contener, obteniendo así una señal puramente alterna. Este paso es esencial, pues en caso contrario también se amplificaría el nivel de continua, anulando completamente la señal.

Si quisiéramos muestrear la señal completa (incluidos los semiciclos negativos) necesitaríamos sumarle a la señal amplificada un nivel de continua, pues el PIC sólo es capaz de detectar tensiones positivas. Esto lo conseguimos presentando en la entrada no inversora (pin 2) una tensión intermedia de 2,5V, aprovechando así al máximo el rango dinámico. Esto puede conseguirse fácilmente con un divisor de tensión en el que ambas resistencias tengan el mismo valor, pero ¿cuál debe ser ese valor

Un amplificador operacional ideal no consumiría corriente a través de sus entradas, pero uno ideal sí requiere una pequeña corriente para alimentar los transistores que hay en su interior. Esta corriente viene especificada en la hoja de datos como corriente de bias, y suele ser del orden de unas pocas decenas de nanoamperios. Las resistencias del divisor de tensión deben elegirse de modo que la corriente que circule por ellas sea de un orden de magnitud superior; en nuestro caso, del orden de las centenas de nanoamperios. Por este motivo hemos elegido dos resistencias de 10K.

Para garantizar la estabilidad de este punto intermedio de tensión frente a interferencias en la alimentación colocamos también un condensador de 22uF a tierra en la entrada no inversora (pin 3).

La introducción se este nivel

En los tiempos que corren, la opción casi forzosa es recurrir a una conexión USB.

Se ha elegido el programador de puerto paralelo GTP Lite en combinación con el software gratuito WinPIC800 para la programación.

de continua en la señal amplificada se ha realizado meramente con fines didácticos, para poner de manifiesto las posibilidades del amplificador inversor con realimentación negativa. Sin embargo, como en nuestra aplicación no nos interesa tanto la señal completa en sí, como su intensidad, y con vista a facilitar el desarrollo del software, nos convendría renunciar al nivel de continua. Para ello deberíamos introducir un condensador entre la salida del amplificador y la entrada del PIC.

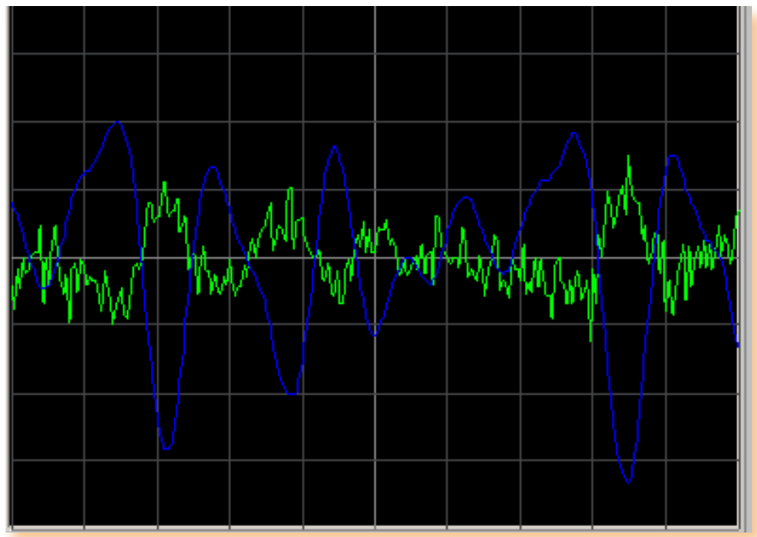
No obstante, hemos decidido seguir un camino diferente, aprovechando el otro amplificador del LM358 para crear un buffer de ganancia unitaria que nos permita alimentar unos auriculares. De este modo podremos comprobar el funcionamiento de la etapa de amplificación aunque no dispongamos de osciloscopio. Esta segunda etapa de amplificación (en realidad, de adaptación) nos permite obtener corriente para alimentar los auriculares sin afectar a la verdadera amplificación (que se realiza en la etapa anterior).

El condensador al que nos referíamos anteriormente lo hemos trasladado a la salida de esta segunda etapa (pin 7), de modo que podremos conectar a continuación de él unos auriculares y comprobar que el micrófono y la amplificación están funcionando correctamente.

En el circuito final sustituiremos la presencia de estos auriculares por una resistencia de 220R que simule una carga, aunque en realidad no sería estrictamente necesario.

En la figura 7 se han representado la señal procedente del micrófono una vez superado el condensador (en verde), y la señal a la salida del condensador de la segunda etapa de amplificación (en azul). La base común de tiempos es 1ms por división, pero la señal original posee una escala de 5mV por división mientras que la amplificada está visualizada a 200mV por división.

II FIGURA 7 II ***Representación de la señal original en verde y de la señal amplificada en azul.***



Puede observarse que la señal amplificada está efectivamente invertida respecto a la original y que su ganancia en ese momento está configurada aproximadamente en 80 a través del potenciómetro.

.Las conexiones del PIC

Además de las conexiones de alimentación y tierra, nuestro PIC necesita un oscilador y una tensión alta en su pin Master Clear Reset (MCLR). Esta tensión se establece a través de una resistencia pull-up de 10K.

Respecto al oscilador, buscando hacer funcionar el PIC a su máxima velocidad (48MHz) colocaremos un cristal de 20MHz entre los pines OSC1 y OSC2, con sus correspondientes condensadores de 15pF a tierra. Durante la programación del PIC indicaremos los valores oportunos de prescaler y postscaler que se aplicarán al PLL (Phase Lock Loop) interno de 96MHz para obtener 48MHz.

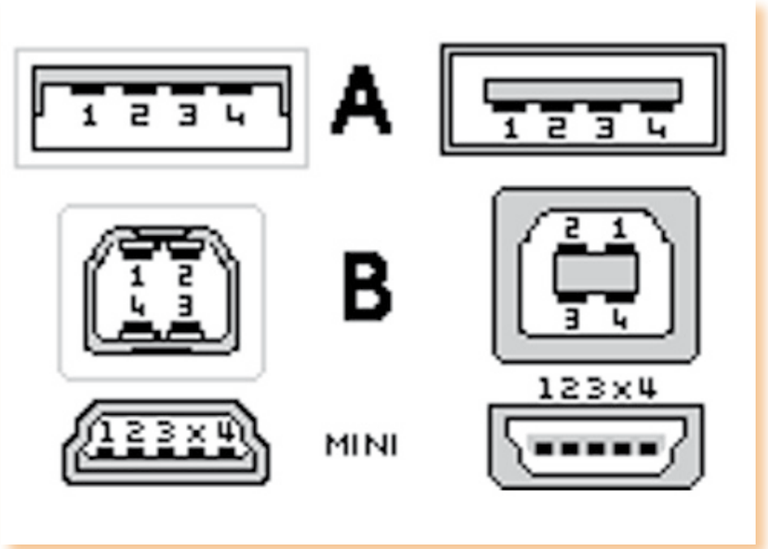
Al realizar las conexiones USB conviene tener presente el tipo de conector que vayamos a utilizar para unir PIC-Tengu al ordenador. Los pines 1 y 4 corresponden a la tensión positiva y la tierra respectivamente, el pin 2 deberá conectarse al pin 15 del PIC (D-) y el pin 3 al pin 16 (D+). La figura 7 le ayudará a identificar la posición de estos pines en los distintos tipos de conectores USB.

Para que la conexión USB sea operativa no debemos olvidar colocar en el pin 14 (VUSB) un condensador de 470nF a tierra.

Las conexiones que hemos establecido hasta ahora son fijas, es decir, no podrían realizarse en pines diferentes del PIC. Por el contrario, las que nos quedan aún (el botón, el conversor ADC y los LEDs de la matriz) podrían realizarse de formas muy variadas, pues la mayoría de los pines pueden cumplir más de una misión y el PIC 18F2455 ofrece varias fuentes de interrupción externa

La señal amplificada nunca superará los niveles impuestos por la tensión de alimentación del amplificador.

II FIGURA 8 II: ***Disposición de los pines en los distintos tipos de conectores USB.***



Un amplificador operacional ideal no consumiría corriente a través de sus entradas, pero uno ideal sí requiere una.

(para el botón) y varias entradas al conversor ADC. La elección de una disposición u otra debe realizarse buscando que el ruteado del PCB resulte lo más sencillo posible. En nuestro caso elegiremos la interrupción 1 (pin 22) para el botón, y el canal 12 (pin 21) para la entrada de la señal procedente del micrófono.

El botón lo conectamos en serie con 2 resistencias de 220R y 10K, de modo que cuando no esté pulsado presente una tensión alta al pin 22, pero que al pulsarse utilice la resistencia de 220R como pull-down, colocando el pin 22 prácticamente a la tensión de tierra.

Como ya decidimos multiplexar los LEDs de la matriz por columnas, deberemos unir los ánodos de todos los LEDs de cada fila y los cátodos de todos los LEDs de cada columna. Los ánodos se alimentarán directamente desde pines del PIC, pero los cátodos se descargarán a través del ULN2803.

Como tenemos pines suficientes, utilizaremos uno para controlar cada sumidero de corriente del ULN2803. Si no hubiera sido así, podríamos haber recurrido a un registro de desplazamiento (como el 74HC595) para controlar todos los sumideros con sólo dos pines del PIC. Además, como nos sobran 2 sumideros del ULN2803, conectaremos a ellos los LEDs que representan los ojos para que la corriente que circula por ellos no tenga que ser canalizada directamente a través del PIC.

Un detalle a tener en cuenta del ULN2803 en la fase de programación es que utiliza lógica inversa, es decir, si colocamos una tensión baja en su entrada, la salida se colocará en alta, bloqueando el paso de la corriente.

Los LEDs rojos que utilizaremos tienen un consumo de 10mA y producen una caída de tensión de aproximadamente 2V, de modo que tendremos que colocar en serie con ellos resistencias de 330R que hagan caer aproximadamente los 3V restantes al ser atravesadas por 10mA. En realidad, el tiempo que van a estar encendidos los LEDs de la matriz (no así de los ojos) es tan breve que podríamos haber prescindido de estas resistencias. No obstante, por si cometiésemos algún error en la fase de programación que los mantuviera encendidos un tiempo mayor, es conveniente tomar la precaución de incluir las resistencias, aunque posteriormente las eliminemos o reduzcamos su valor para lograr mayor brillo. Tenga en cuenta que si decide utilizar LEDs de otro color tendrá que calcular convenientemente el valor de estas resistencias de acuerdo a su correspondiente corriente y caída de tensión; particularmente, los LEDs azules suelen funcionar con una tensión de 3,4V. ■

Y a continuación...

En la siguiente entrega de este proyecto presentaremos dos diseños del PCB, uno para componentes through-hole y otro para componentes de montaje superficial, y describiremos minuciosamente el software que dará vida a nuestro pequeño PIC-Tengu.

PIC Trainer 2.0

Te proponemos la construcción de tu propia placa entrenadora para microcontroladores PIC. Con este sistema, basado en diferentes módulos, podrás aprender todo lo que necesitas sobre el funcionamiento de estos circuitos integrados, sin necesidad de gastar una pequeña fortuna.

//por:Ariel Palazzesi//
arielpalazzesi@gmail.com

Si recién estas dando tus primeros pasos en el mundo de los microcontroladores PIC, es posible que te encuentres ligeramente abrumado por la cantidad de conceptos que debes asimilar para poder desarrollar tus proyectos.

Esto es inevitable, ya que para que un proyecto basado en un microcontrolador no sea un fracaso, debemos dominar tanto el diseño del circuito electrónico como la confección del programa que se encargará de hacerlo funcionar.

La idea detrás de una placa "entrenadora" o "trainer" es disponer de una herramienta que nos aporte un hardware probado, confiable, que podamos utilizar cuando queremos probar alguna pieza de software, sin necesidad de construir un circuito especial para ello. Podemos ver a un entrenador como un "proyecto universal", que dispone de prácticamente todos los elementos que necesitamos para probar nuestro software.

El uso de jumpers permite configurar los módulos.

.El PIC TRAINER

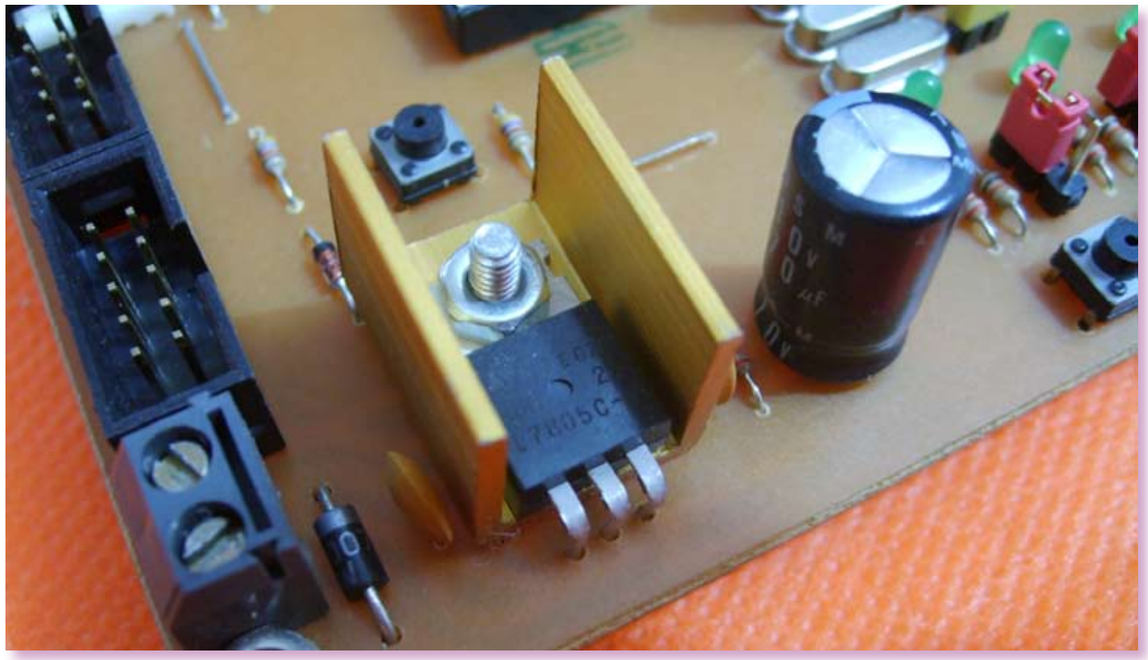
Hace un par de años, cuando comencé a interesarme por la electrónica y los microcontroladores, rápidamente me di cuenta que disponer de un entrenador podría ahorrarme bastante tiempo, y evitar frustrarme cada vez que diseñaba un PCB para probar alguna rutina de software y que por algún motivo no funcionaba.

Existen algunos trainers disponibles comercialmente que son realmente impresionantes. Disponen de un muy buen numero de dispositivos on-board (desde pulsadores y LEDs hasta pantallas LCD graficas), y ofrecen soporte para un numero de microcontroladores increíblemente numeroso. A veces, incluso soportan chips de diferentes fabricantes.

Pero me pareció que podría aprender bastante más si diseñaba mi propia placa de entrenamiento. En ese momento era incapaz de fabricar un PCB de doble cara (creo que ni siquiera hubiese podido dibujarlo), por lo que debí sacrificar unas cuantas funciones que inevitablemente reque-

Quizás la característica más sobresaliente de este proyecto sea su modularidad.

Algunos módulos disponen de su propio regulador de voltaje.



Existen algunos trainers disponibles comercialmente que son realmente "impresionantes".

rían de un PCB demasiado complejo.

Cuando ya tenía un diseño razonablemente funcional (aunque implicaba una placa de unos 20x25 centímetros) y que posiblemente hubiese funcionado, se me ocurrió que podría ser mucho más eficiente y sencillo dividir el entrenador en una serie de módulos. De esta manera, podría ir construyéndolos a medida que los necesitaba, sin necesidad de gastar mucho dinero o tiempo de una sola vez.

Así fue como nació el PIC TRAINER, del cual analizaremos su "segunda reencarnación", la versión 2.0.

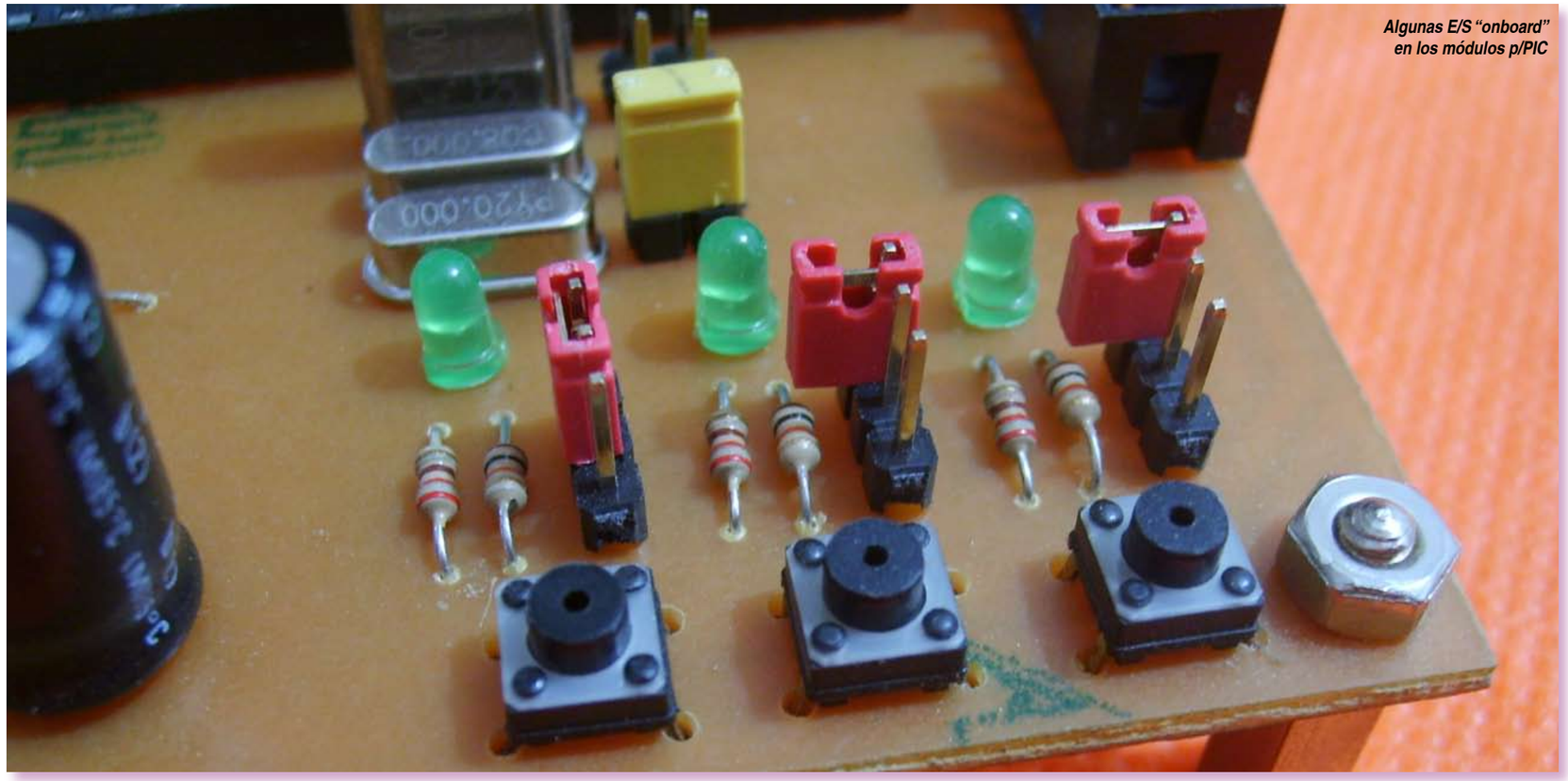
.Características

Quizás la característica más sobresaliente de este proyecto sea su modularidad. En efecto, se ha dividido el entrenador en módulos, cuatro de ellos destinados a alojar los microcontroladores de 8, 18, 28 y 40 pines y el puñado de componentes que necesitan para funcionar. El lector puede construir solo uno de ellos (el que soporte el modelo de PIC que desea estudiar) o bien los cuatro.

Estos módulos centrales, a los que hemos bautizado como PIC TRAINER 8, PIC TRAINER 18, PIC TRAINER 28 y PIC TRAINER 40, disponen de un zócalo para alojar al microcontrolador en cuestión; un pulsador destinado a, en caso de ser necesario, efectuar el RESET del mismo; tres cristales seleccionables mediante jumpers, para utilizar diferentes frecuencias de trabajo; un regulador de voltaje (y componentes asociados) dedicado a brindarle al PIC los niveles de tensión y corriente que necesita para funcionar; un conector para la programación ICSP; y algunos pulsadores y LEDs que proporcionan unas pocas líneas de entrada/salida.

Todo lo demás se ha separado en módulos accesorios, que se conectan al modulo central elegido mediante cables planos de 10 vías terminados en conectores IDC. He realizado una buena cantidad de módulos: displays LCD 2x16, 8 entradas y salidas, 4 u 8 relés, RS-232, EEPROM, I2C, etc.

Todos los módulos diseñados hasta la fecha utilizan PCBs de una sola cara, lo que hace muy fácil su fabricación.



Algunas E/S "onboard" en los módulos p/PIC

Todos los módulos disponen de conectores compatibles eléctricamente, para poder intercambiar los cualquier modulo de expansión.

.Cables y conectores

Todos los módulos disponen de conectores compatibles eléctricamente, lo que permite, al menos en teoría, utilizar cualquier modulo de expansión con cualquier modulo central.

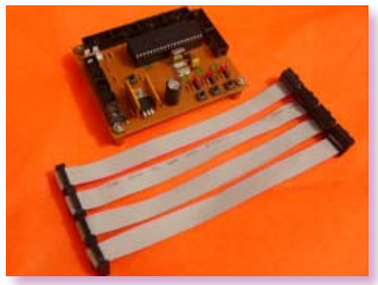
Uno de los temas más importantes a tener en cuenta en el desarrollo de un sistema modular es la estandarización de los cables y conectores, de forma que sea prácticamente imposible conectar por error algo donde no se debe, dañando algún componente.

Hemos decidido utilizar conectores IDC10, que poseen 10 vias. Y de los 10 pines disponibles sólo se utilizan 6. Esto obedece a varias razones. Por un lado, el no emplear 4 de los pines que quedan del lado de "afuera" de la placa permite que el diseño de los PCB sea mucho más sencillo, sin tener necesidad de pasar con una pista por entre otros dos pines. De esta manera, se puede utilizar pistas anchas, apropiadas para aquellos que realizan sus placas de circuito impreso con el "método de la plancha" descrito en el número 1 de la revista.

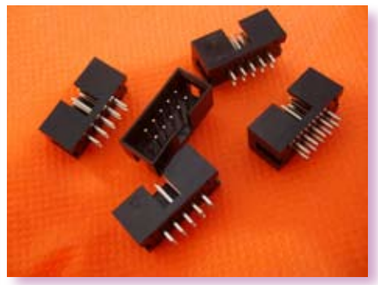
Por otra parte, el hecho de utilizar solo 4 líneas de datos nos permite aprovechar mejor los pines de cada puerto del PIC. Efectivamente, si se hubiesen utilizado los 8 pines de cada puerto en cada conector, en aquellos casos en que un modulo hiciera uso de uno o dos bits de datos (por ejemplo, el modulo RS-232) nos quedarían 6 pines del puerto elegido inutilizados, ya que no podemos insertar dos fichas a la vez en el conector.

Como contraparte, algunos

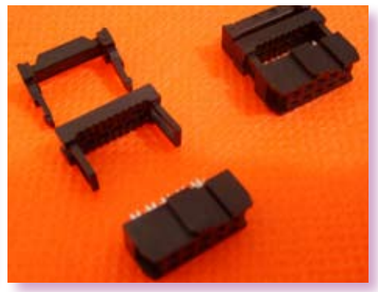
Podemos ver a un entrenador como un "proyecto universal", que dispone de casi todos los elementos que necesitamos para probar nuestro software.



PIC TRAINER 40 y algunos cables de conexión ya armados"



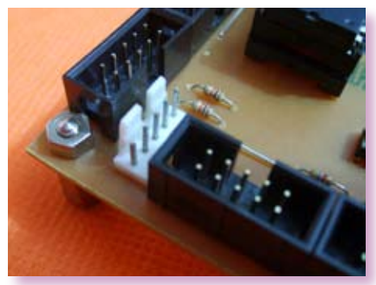
Conectores IDC10 utilizados en todos los módulos.



Estas son las fichas que deben montarse en los cables planos.

Forma en que deben montarse los cables.

Cables montados. Los extremos son simétricos.



|| FIGURA 1 || *Las placas centrales poseen conexión ICSP.*

módulos deben ser enlazados con más de un cable, aunque este es un mal menor.

La imagen del conector nos muestra la función de cada uno de los 10 pines que posee. En general, todos los módulos tienen los conectores montados sobre los bordes, de forma que los cables de conexión puedan colocarse fácilmente. La fila de pines etiquetados **"NC"** queda hacia el lado de afuera del PCB, y la ranura presente en la ficha macho hacia adentro.

Los cuatro pines "**NC**" son los que no están conectados. Los identificados como **D0**...**D3** corresponden a las líneas de datos, y he intentado (con bastante éxito) que se correspondan a los pines **0**...**3** o **4**...**7** de cada puerto de cada PIC. Por ultimo, "**+5V**" y "**GND**" proporcionan alimentación eléctrica a muchos de los módulos.

Como esta corriente proviene de la placa del módulo central al que se hayan conectados, y es suministrada mediante un regulador de voltaje integrado LM7805 que sólo puede proporcionarnos (teóricamente) 1 Amper, debemos ser cuidadosos con los consumos. Esta es la razón de que algunas placas tengan, además, una bornera para proveer su propia alimentación, de forma de evitar recargar el regulador de voltaje de la placa principal.

La construcción de los cables no podría ser más sencilla. Dado que los conectores IDC son "crimpleables" (es decir, pueden aplicarse al cable simplemente presionándolos fuertemente contra él), solo nos tomara unos minutos disponer de un buen número de ellos.

Tenemos que recordar, al ahora de colocar las fichas sobre el cable plano, que este debe quedar armado con sus extremos de forma simétrica (ver fotos). Esto permitirá que sus extremos, a la hora de insertarlos en los módulos, puedan intercambiarse sin problemas. De esta manera resultará prácticamente imposible dañar algún módulo por haber insertado un cable de forma incorrecta. La "muesca" que tiene cada conector impide que al ficha entre en una posición que no sea la adecuada.

.ICSP

Las placas destinadas a albergar los microcontroladores (como PIC TRAINER 8, PIC TRAINER 18, PIC TRAINER 28 y PIC TRAINER 40) como adelantamos, poseen un conector para la programación ICSP (In Circuit Serial Programming, o Programación Serial en circuito).

El ICSP es el sistema utilizado en los dispositivos PIC de Microchip para programarlos sin necesidad de tener que retirar el chip del circuito del que forma parte. Esta forma de programación es válida para todos los PIC de la gama baja, como puede ser el 12C508 y 12F629; los de la gama media, desde el de 16F84A hasta el 16F877A; e incluso para la familia 18xxxx.

Básicamente, se trata de un sistema de programación serie síncrona en el que intervienen 2 señales: una de entrada y salida para la transmisión y recepción de datos; y otra de entrada para la sincronización de la transmisión y recepción de los datos. Las líneas utilizadas se ubican (generalmente) en el pin RB6/PGC para la señal de sincronización (reloj) y en el RB7/PGD para los datos.

El protocolo trabaja con dos tensiones, una de alimentación (VDD), cuyo rango de valores está comprendido entre 4.5 y 5.5 voltios, y otra de programación (VPP) cuyo rango oscila entre un mínimo y un máximo de 12 y 14 voltios respectivamente.

Los programadores generalmente poseen un conector que entrega estas señales, y las placas que hemos diseñado tienen el conector correspondiente para recibirlas. Queda a cargo del lector la tarea de construir un cable que vincule el programador con el entrenador.

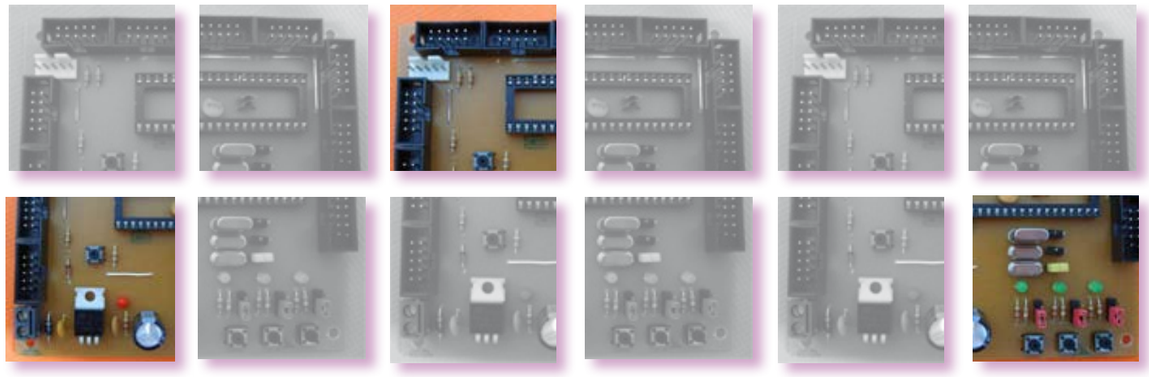
El pinout correspondiente al conector ICSP de todas las placas que forman parte del PIC TRAINER es el mismo, y puede verse en la figura correspondiente. (Ver Figura 1)

.Módulos disponibles

En realidad, y a partir de las especificaciones publicadas para cada conector, es posible crear módulos de expansión limitados solo por la imaginación o necesidad de cada lector. Nosotros hemos diseñado modulos centrales para PICs de cualquier número de pines, ampliación de entradas y salidas, reles, LCD alfanumérico y LCD gráfico, teclados, conexión RS-232, enlaces por RF, etc.

La idea es ir publicando un puñado de ellos en cada numero de la revista uControl. También puedes encontrar módulos e información adicional en nuestra página web: www.ucontrol.com.ar ■

módulo PIC TRAINER 40



El primer módulo de nuestro entrenador es prácticamente indispensable. Se trata de unos de los cuatro que poseen capacidad para albergar un microcontrolador. Concretamente, es el encargado de albergar a los microcontroladores de 40 pines en formato DIP, tales como el PIC16F877A, PIC16F887A y muchos más.

//por:Ariel Palazzesi//
arielpalazzesi@gmail.com

Afortunadamente Microchip coloca los puertos de los micros de 40 pines casi siempre en el mismo lugar, lo que permite a esta placa la posibilidad de ser utilizada con diferentes modelos, incluso con algunos de la serie 18F, tales como el PIC18F4525, PIC18F4620, PIC18F442 o PIC18F452.

Si miras las hojas de datos correspondientes, seguramente encontraras que muchos microcontroladores más pueden funcionar en este módulo. Incluso, es posible construir un "adaptador" para poder utilizar en él micros con capsula LQFP.

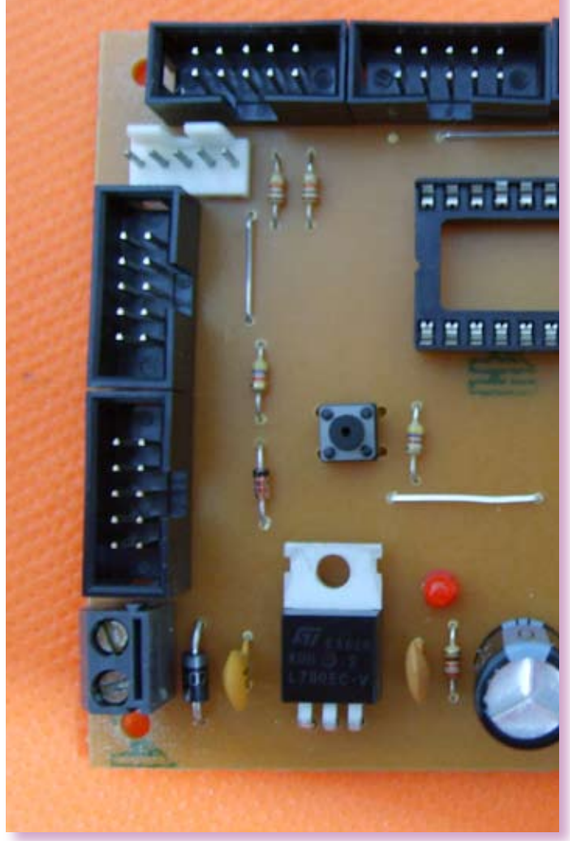
.El circuito

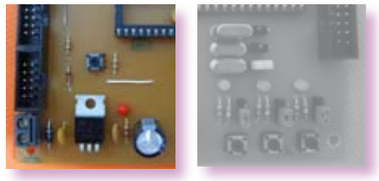
Como puede verse en el diagrama que acompaña este artículo, el circuito de este modulo es bastante simple. Básicamente, se trata de "publicar" los pines correspondientes a los puertos de entrada y salida del microcontrolador alojado en el zócalo central mediante una serie de conectores IDC10.

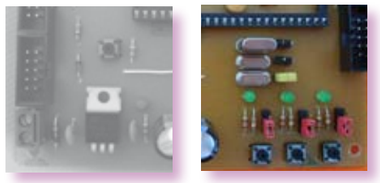
La alimentación del modulo se ha resuelto mediante un regulador de voltaje integrado de la serie LM78xx, concretamente el modelo LM7805. Una bornera de dos tornillos permite alimentar a la placa desde una fuente externa de corriente continua con una tensión de entre 7.5 y 15V. Un diodo 1N4007 se encarga de proteger el resto de la etapa de alimentación de una conexión con la polaridad invertida.

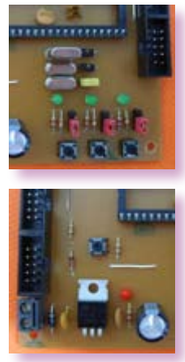
El regulador de voltaje esta dotado de los dos condensadores de 0.1 uF de rigor, y a la salida un condensador electrolítico de 470uF/16V se encarga de eliminar el ripple que pudiese encontrarse a la salida de la etapa de alimentación. Por ultimo, un diodo LED, en serie con un resistor de 220V se enciende cuando el circuito está alimentado, indicando esta condición.

En lugar de utilizar un cristal como oscilador del PIC que está siendo utilizado en el módulo, hemos colocado 3. Una serie de jumpers (identificados como JP4, JP5 y JP6) se encargan de seleccionar uno de ellos. Los dos condensadores de 22pF completan esta parte del circuito. Aunque parezca obvio, tenemos que recordar al lector que no debe









Este es uno de los módulos de nuestro entrenador que podríamos calificar de indispensable.

colocar más de un jumper a la vez, ya que en ese caso el microcontrolador no funcionará.

Hemos elegido para nuestro prototipo valores de 4MHz, 8 MHz y 20MHz, pero nada impide utilizar otros valores. El lector puede cambiarlos a gusto.

En caso de utilizar algún modelo de PIC que disponga de oscilador interno y se quieran utilizar los pines 13 y 14 del mismo (correspondientes, en general, a E3 y E4) como pines de entrada/salida, bastará con no colocar ninguno de los jumpers mencionados.

En el caso de configurar los pines 13 y 14 como entrada/salida, estos se comportarán de la misma manera que el pin 10, correspondiente al bit 2 del PORTE. Esto habilita los LEDs y pulsadores incorporados en el modulo, permitiendo su uso como forma de ingresar (o representar) datos a (o de) nuestro programa. Los jumpers JP1, JP2 y JP3 permiten seleccionar si conectamos el LED o el pulsador al PIC.

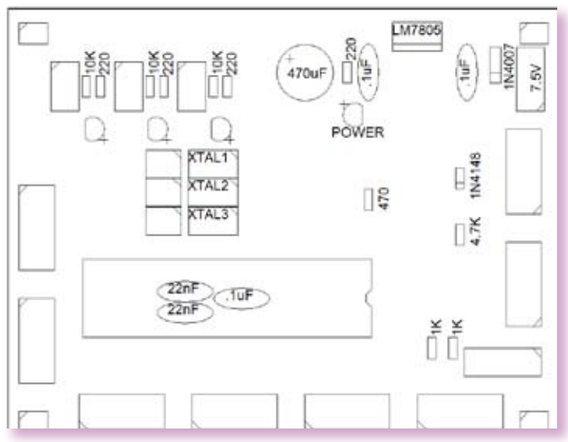
En caso de seleccionar los pulsadores, debemos recordar que estos ponen el pin correspondiente a 5V cuando son presionados. Mientras que están en reposo, las entradas se mantienen a GND a través de sendos resistores de 10K.

Los conectores siguen las mismas normas que explicamos en el artículo anterior, así que no deberías tener problemas a la hora de determinar la función de cada pin. Como regla general, recuerda que de los pines exteriores de cada conector solo se emplea uno (+V) y los otros cuatro están sin conectar. De lo cinco interior, uno corresponde a GND y los otros 4 a datos. No es mala idea tener a mano el grafico con la función de cada pin a mano cuando decidas hacer algún programa para cargar en el entrenador.

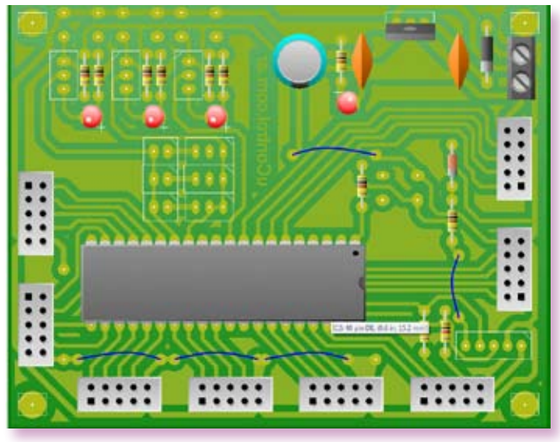
La única excepción, o desvío de lo normal, que puedes ver en el diagrama de los conectores es en los pines correspondientes a RB6 y RB7, ya que poseen un resistor de 220 ohms en serie. Cumplen con la función de permitir programar el PIC mediante el conector ICSP sin necesidad de retirar el cable que conecta el entrenador con el modulo de turno. Por supuesto, si lo deseas puedes reemplazar esos dos resistores por sendos puentes, y a otra cosa. Solo deberás quitar el cable plano a la hora de reprogramar el PIC.

El pin 1, que corresponde al RESET en los microcontroladores PIC de 40 pines (al menos, en los que son compatibles con este entrenador), esta unido a un pulsador a través de un resistor de 470 ohms y a +V mediante otro de 4.7K y un diodo 1N4148. Al presionar el pulsador, el microcontrolador se resetea. Durante el funcionamiento normal del programa, el pin esta a +V. El diodo impide los problemas que podrían surgir entre las alimentaciones del modulo y del programador al utilizar el conector ICSP.

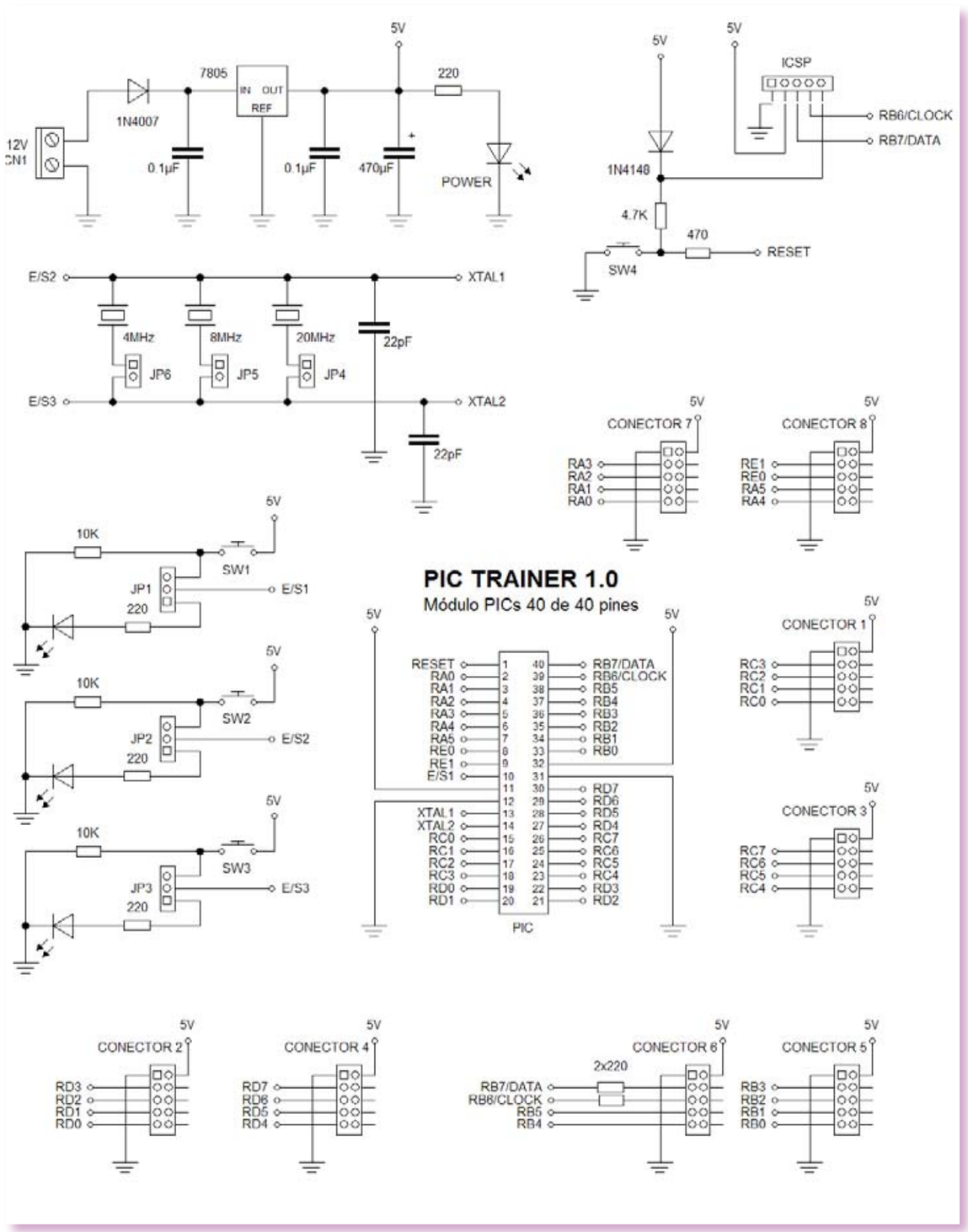
Esta imagen ayudará a colocar los componentes sobre el PCB.



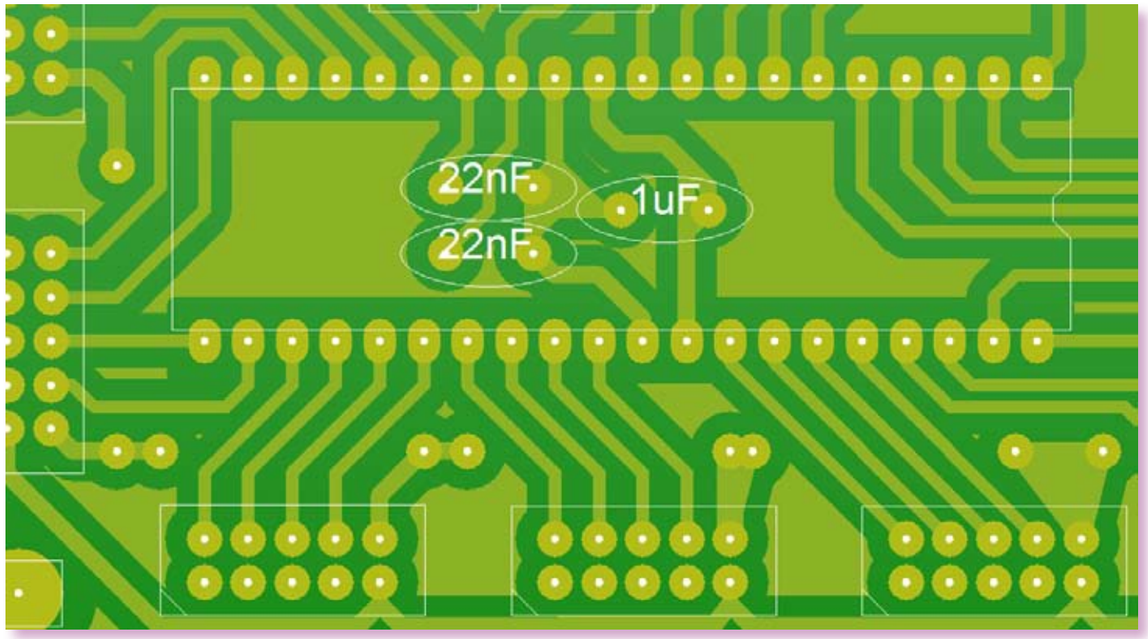
Este es, según el programa de diseño, el aspecto que tendrá en módulo.

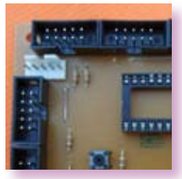


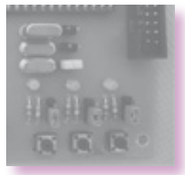
Este es el circuito del modulo PIC TRAINER 40.



Debajo del PIC se ubican 3 condensadores cerámicos.





Los jumpers JP1, JP2 y JP3 permiten seleccionar si conectamos el LED o el pulsador al PIC.

.Construcción

Si ya has montado alguno de nuestros proyectos, no tendrás ninguna dificultad a la hora de construir tu propio entrenador. Descarga el archivo PDF correspondiente al PCB desde nuestra Web, y mediante la forma que más te guste (puedes usar el "método de la plancha" explicado en la revista numero 1) transfiérelo a un trozo de PCB virgen. Luego, al baño de cloruro férrico; y por ultimo, una buena limpieza y haces los agujeros.

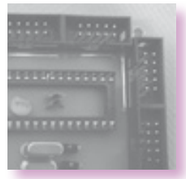
A la hora de soldar los componentes, como siempre, resulta más sencillo si primero vas colocando los que son más bajos, como los puentes, diodos, zócalos y resistores. Deja para el final los conectores, regulador de voltaje y condensadores. Asegúrate de que, involuntariamente, no haces un puente entre dos puntos del circuito.

Hay tres condensadores cerámicos que quedan dentro del zócalo de 40 pines, y debajo del microcontrolador. No es una posición muy bonita para ellos, pero fue la única que encontré a la hora de diseñar el circuito impreso. No molestan a la hora de poner el PIC, así que no hay problemas con eso.

Presta especial atención a la hora de soldar los componentes que tienen "polaridad", como los diodos, LEDs y condensadores electrolíticos. También es importante que coloques el zócalo destinado al PIC en la dirección correcta, ya que de hacerlo mal puedes confundirte cuando insertes el microcontrolador, dañándolo.

Una vez montado todo, sin colocar el PIC en su lugar, alimenta el circuito con una tensión de entre 7.5 y 12V. El LED "Power" debería encenderse. Si es así, verifica con un multímetro que la tensión entre los pines 11 y 31 (o 32 y 12) del zócalo del microcontrolador sea de 5V. También puedes verificar que en los conectores de expansión este presente esa tensión. Si todo esta bien, ya tienes listo tu entrenador. Caso contrario, repasa las soldaduras y posición de los componentes.

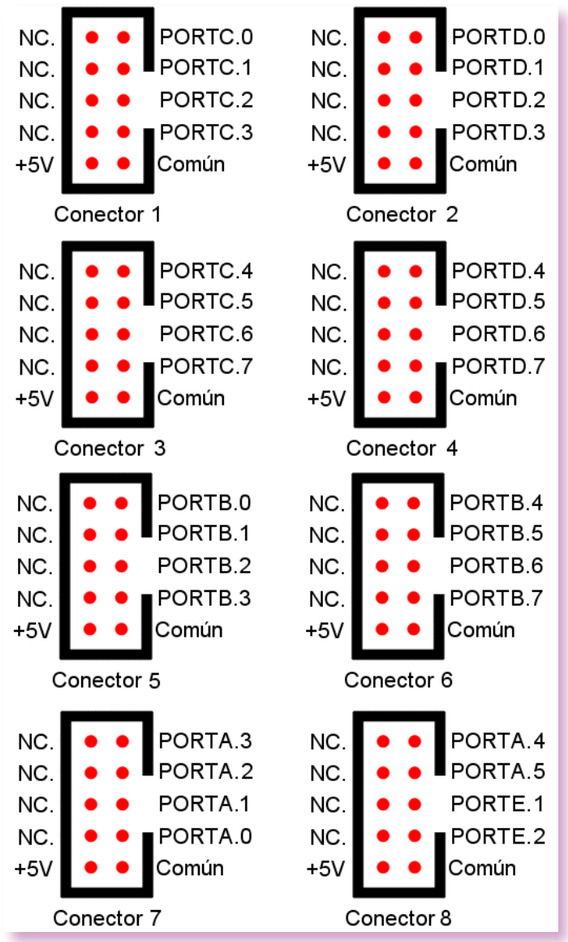
Hay tres condensadores cerámicos que quedan dentro del zócalo de 40 pines, y debajo del microcontrolador.



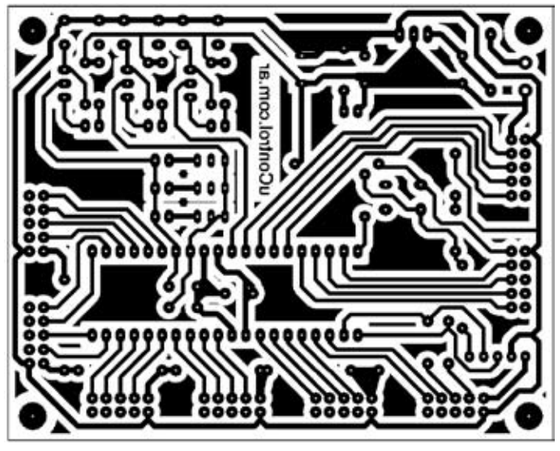
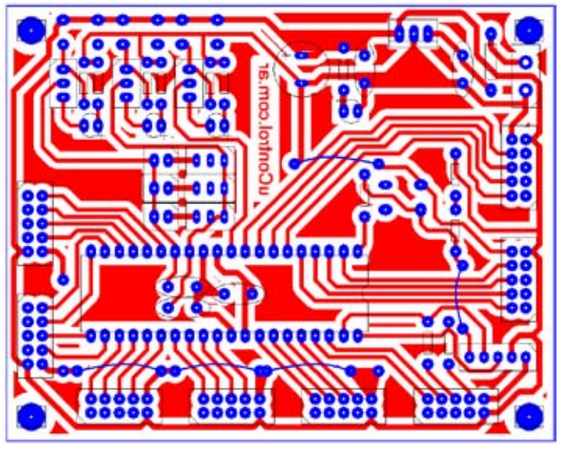
.Conclusión:

Hemos montado la primera placa que componen este entrenador. Personalmente, es el que mas utilidad me ha dado. Por supuesto, por si solo resulta casi inútil, ya que las funciones de entrada/salida implementadas "onboard" son mínimas. Pero si construyes, al menos, el módulo de 8 E/S, tendrás diversión para rato. ■

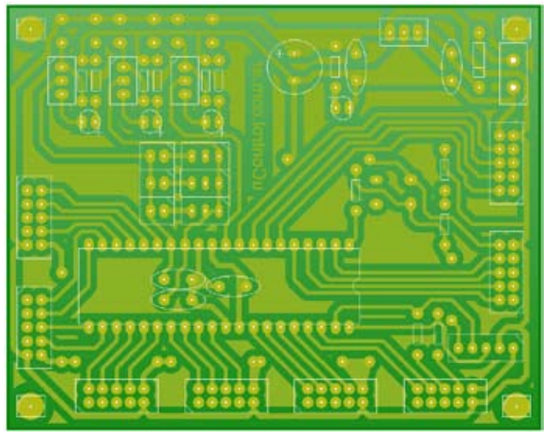
Pinout de cada uno de los conectores de expansión.



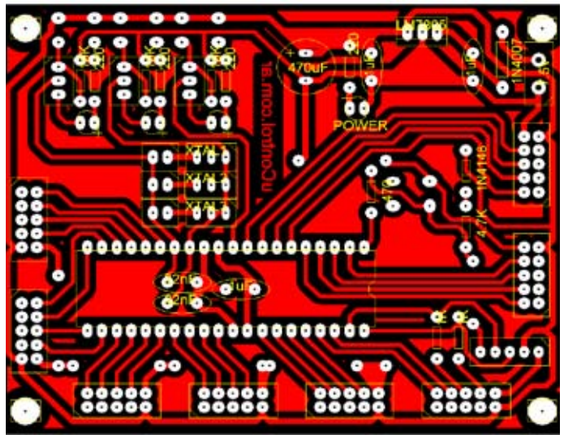
El PCB a utilizar, que mide 98x80 milímetros.

Aspecto de la serigrafía del PCB a construir.



Hemos decidido utilizar un PCB de una sola cara.



PIC TRAINER 2.0: Módulo 8 E/S

Esta es, quizás, una de las placas de nuestro entrenador que deberíamos construir primero. Es indispensable para nuestro entrenador. Consiste en una serie de pulsadores y diodos LEDs, que pueden ser configurados para su uso como dispositivos de entrada o salida, y conectarse a cualquiera de las placas centrales, que contienen los micros.





//por:Ariel Palazzesi//
arielpalazzesi@gmail.com

Para ensamblar esta placa no hay ninguna recomendación en especial.

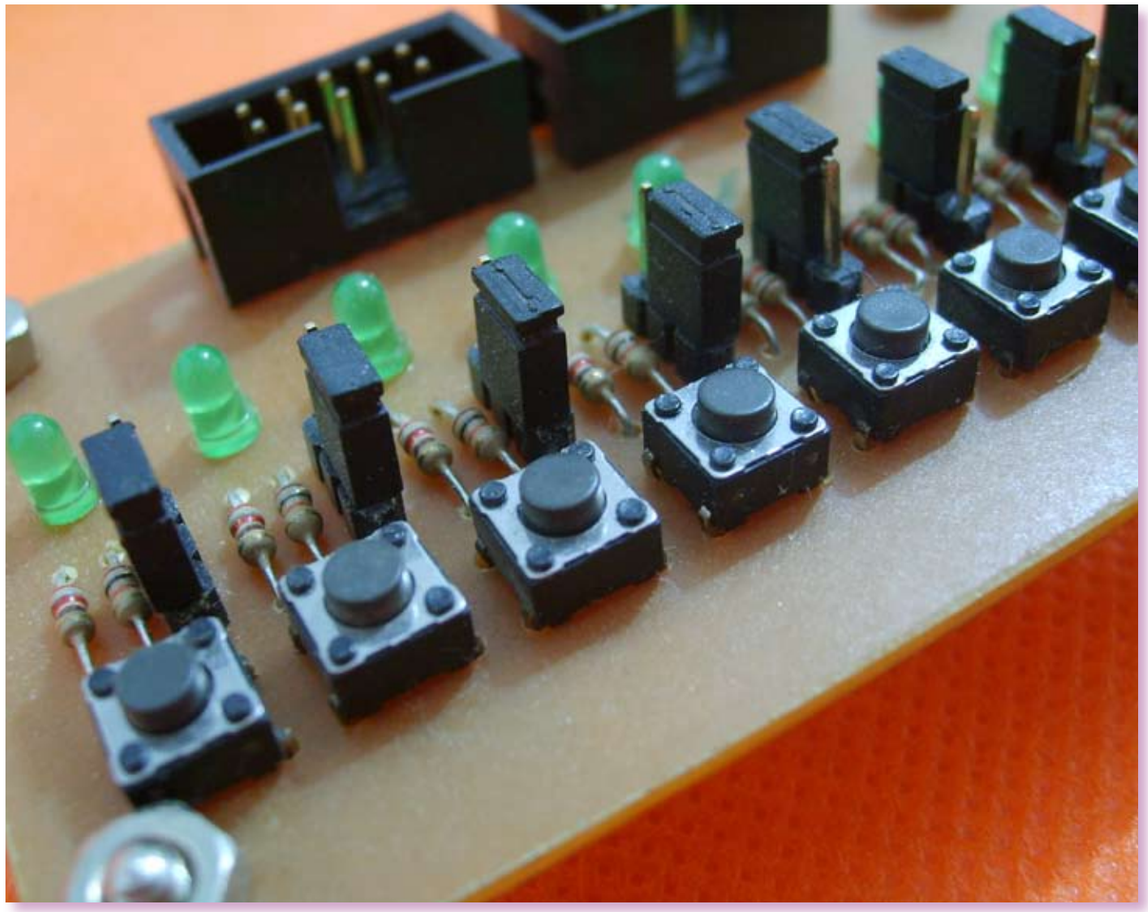



Esta pequeña placa contiene 8 pulsadores y 8 LEDs, pudiendo elegirse mediante 8 jumpers que combinación de ellos vamos a usar. De esta manera es posible, por ejemplo, utilizar 2 líneas como entradas y 6 como salidas, o cualquier combinación que desees.

Su tamaño es "1/2", es decir, la mitad de las de tamaño completo. Mide sólo 49x80 milímetros. La construcción es muy sencilla, puede realizarse en una hora o poco más de trabajo, y su costo es muy bajo. Veámosla en detalle.

.El circuito:

Si bien los módulos que utilizamos para albergar los PIC de 8, 18, 28 o 40 pines disponen de algún pulsador y/o LEDs "on-board", su cantidad seguramente será insuficiente en la mayoría de los casos. Es por ello que este módulo resulta casi indispensable para nuestras prácticas. Por supuesto, si 8 E/S resultan insuficientes, nada impide que construyamos varias placas iguales. Al fin y al cabo, la modularidad es la principal fortaleza de nuestro entrenador.

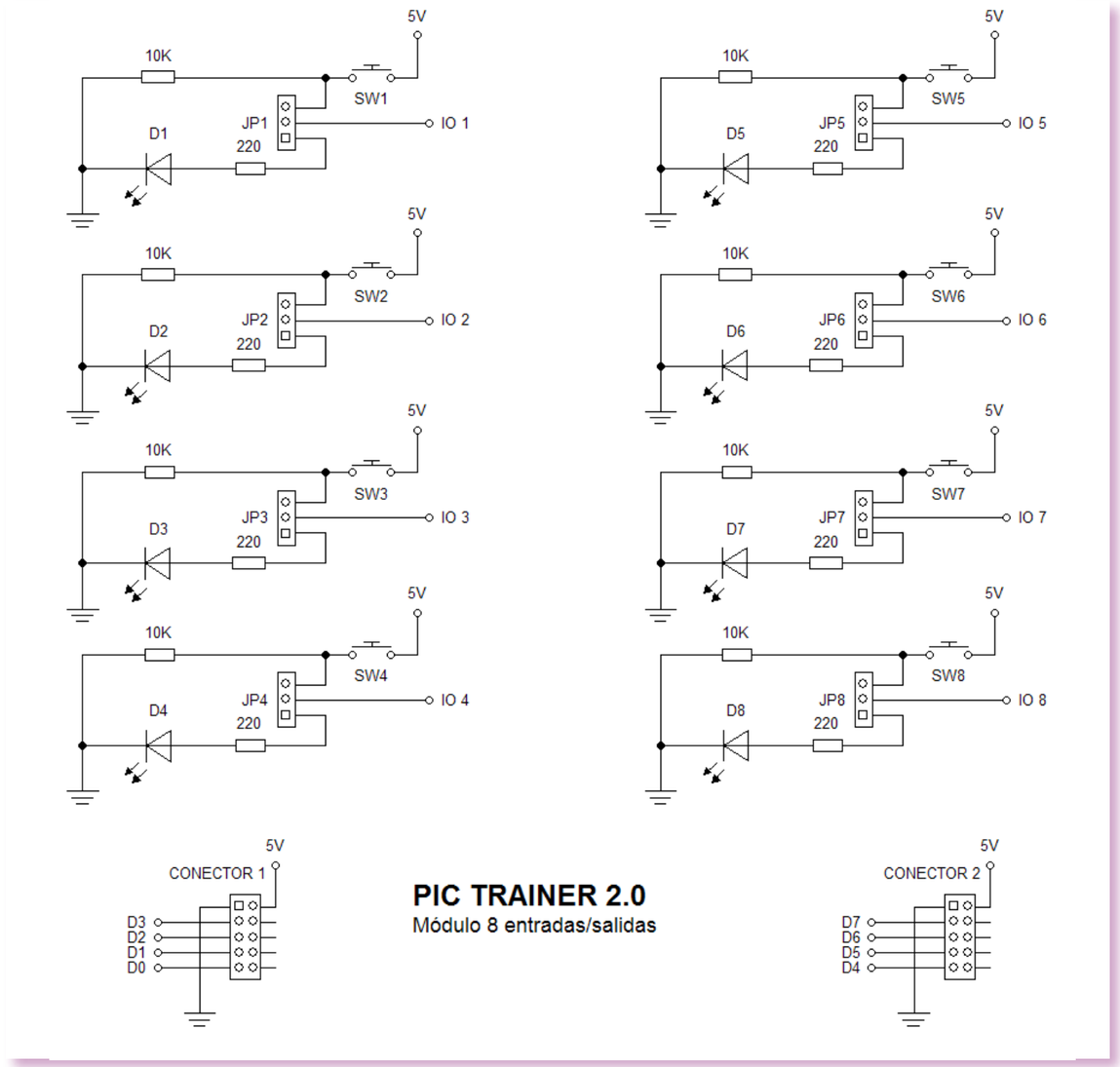








Este es el esquema eléctrico del modulo de 8 entradas y salidas.



Podemos utilizar todos los canales como entradas, como salidas, o dividirlos en grupos.

Viendo el esquema eléctrico del circuito, lo primero que nos llama la atención es que esta placa carece de fuente de alimentación. Efectivamente, este módulo toma la energia que necesita para funcionar directamente del módulo central al que este conectado, consumiendo unos 100 mA cuando los 8 LEDs están encendidos. Esta corriente puede ser provista sin problemas por cualquiera de los módulos centrales previstos.

El circuito consiste en dos bloques iguales, con cuatro LEDs y cuatro pulsadores para circuito impreso (del tipo “normalmente abierto”) en cada uno de ellos.

El hecho de que estén separados en dos bloques obedece a que los conectores de expansión previstos en las placas centrales de este juego de módulos, como vimos, proveen conexión para cuatro pines de los puertos cada una, por lo que se necesitan dos cables de conexión para utilizar los 8 “canales” de entrada/salida. Por supuesto, nada impide usar solo un cable si solo necesitamos cuatro líneas de E/S.

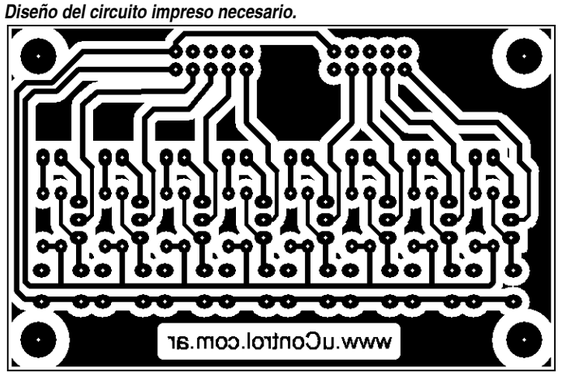
La disposición de pines de los dos conectores IDC10 utilizados respeta las reglas establecidas para los módulos de este entrenador.

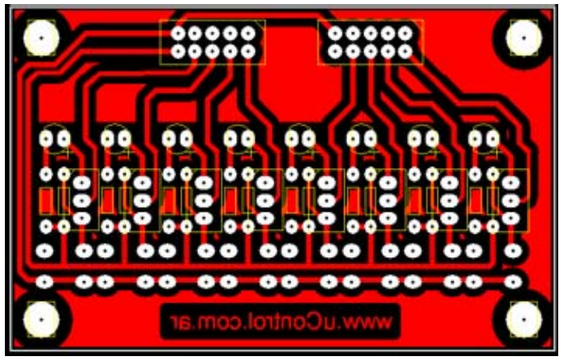
Como se puede ver en las fotos que acompañan este articulo, cada canal de entrada/salida posee un “jumper” (o puente) que se coloca en un grupo de tres pines. Su función es seleccionar si en ese canal se va a utilizar el pulsador (entrada) o el LED (salida). De los tres pines de bronce en que se coloca el puente, el centralr es el “común”, el que esta del lado del pulsador es el que configura el canal como entrada, y el que esta del lado del LED es el que permite utilizar el canal como salida.

Con el puente puesto hacia el lado del LED, la corriente entregada por el pin del microcontrolador circula a través del resistor de 220 ohms, pasando por el LED y haciendo que este se ilumine. Con el puente colocado hacia el lado del pulsador, la corriente que proviene de los 5V de la placa central (a través del cable plano) llega al pin correspondiente del microcontrolador. Para que el pin no quede “flotando” sin conexión cuando el pulsador esta abierto, se uso una resistencia de 10 K puesta a masa (a través del cable plano) en cada canal.

Cualquier combinación es posible: podemos utilizar todos los canales como entradas, como salidas, o dividirlos en grupos de cualquier manera, simplemente poniendo cada jumper en la posición deseada.

Diseño del circuito impreso necesario.





Vista del pequeño módulo terminado.

Lista de componentes:

2 conectores IDC (machos) de 10 vías
24 pines de bronce
8 LEDs de 3mm
8 resistores de 220 ohms 1/4 de watt
8 resistores de 10 Kohms 1/4 de watt
8 pulsadores normales abiertos para impreso (de 5 mm)
Pertinax, jumpers, etc.

El circuito consiste en dos bloques iguales, con cuatro LEDs y cuatro pulsadores en cada uno de ellos.

.El circuito impreso

El tamaño de la placa es muy reducido, y como dijimos, solo ocupa unos 49x80 milímetros. Esto obliga a utilizar LEDs de 3mm de diámetro en lugar de los más comunes de 5mm.

Para ensamblar esta placa no hay ninguna recomendación en especial. Como siempre decimos, utilizamos una impresión láser del circuito impreso propuesto y con el calor de una plancha común lo transferimos al cobre. Si tienes dudas de cómo realizar este proceso, puedes consultar el artículo al respecto en el número 1 de la revista.

Una vez que tenemos la placa lista y agujereada, procedemos a soldar los componentes, cuidando de que los LEDs estén orientados correctamente, si no, no encenderán.■

Reloj de Tiempo Real (RTC)

El Relojito es el proyecto ideal para utilizar a este pequeño gigante. Básicamente, el DS1307 se encarga de mantener funcionando un reloj de tiempo real, extremadamente preciso, al que podemos consultar desde nuestro programa cuando lo deseemos. En este articulo, Diego Márquez García-Cuervo, hace un divertido análisis de su funcionamiento.

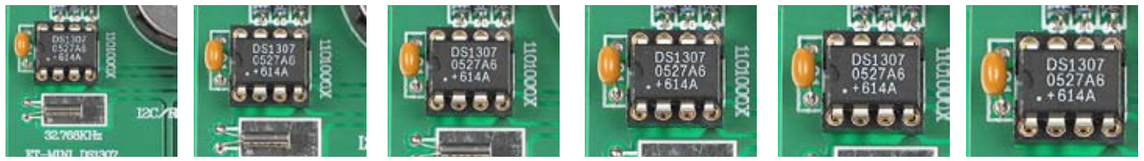
//por:Diego Márquez García-Cuervo//

El DS1307 tiene un pin de salida que, debidamente habilitado, nos ofrece una onda cuadrada.

Nuestros PIC's no saben en qué día y hora viven. Perdidos en el no-tiempo, la realidad analógica o digital es un continuo donde las fracciones de tiempo superior a algunos milisegundos son entes abstractos e inasibles. Anclados en patrones de frecuencia decimales que no son divisibles por sus primos binarios, esclavos del redondeo, siempre les sobran o les faltan unos microsegundos para dar el Segundo Perfecto. Como Reloj en Tiempo Real, un PIC abandonado a su cuarzo y a sus divisores, preescalers y postescalers, simplemente no da la talla.

Y ahí es donde entran los amables señores de Dallas Maxim y su cucaracha octópoda clockeadora: el DS1307, intitulada por ellos mismos como un "64 x 8, Serial, I2C Real-Time Clock", o sea, un corazón de reloj con alguna RAM adicional (no volátil ya que es asistida por una batería de Litio que la mantiene a flote mientras no le falte el fuelle de los voltios).

Este artículo consiste esencialmente en exponer la información necesaria para que nuestros PICs, aprovechando que ambos hablan en I2C, puedan trabajar juntos.



Block Diagram

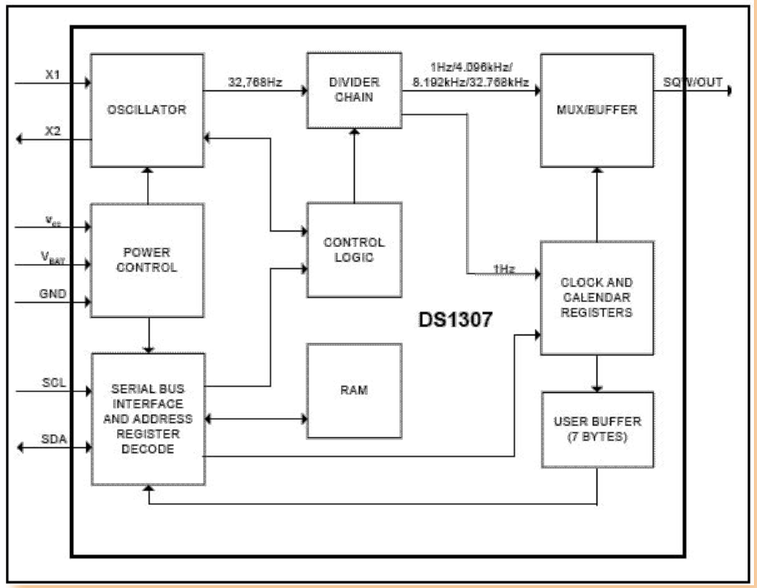


Diagrama en bloques del DS1307 (Dallas Maxim)

.Implementación

La implementación consiste en posibilitar la comunicación I2C entre nuestros proyectos y el **DS1307**, explicando lo fundamental de este chip, mostrándonos a continuación qué podemos hacer con él y qué no (visto desde el punto de vista de dentro de un PIC claro está, ya que es de cajón y no pienso comentar por ejemplo que tragarse un DS1307 sin quitarle las puntiagudas patillas puede ser incompatible con la vida, tal y como la conocemos).

No pienso repetir palabra por palabra, como un vulgar loro de bar portuario, lo que ya dice de forma meridianamente clara el Datasheet del DS1307, así que tras poneros el Diagrama de Bloques del mismo (extraído de dicho datasheet) paso a comentar algunas circunstancias interesantes a tener en cuenta.

un pequeño y econmico integrado, que puede ayudarnos mucho en nuestros proyectos.

1º El cristal de cuarzo ha de ser de 32.768 KHz. Un arcano designio que tiene que ver con que 2^15 = 32.768, por lo que esa frecuencia es divisible de forma exacta binariamente para generar 1000, o sea, nuestro segundo perfecto.

2º La alimentación es doble. Por un lado el VCC de nuestro circuito normal, el del PIC, y por otro una batería de Litio, que va a permitir que el reloj siga su normal funcionamiento aún cuando apaguemos el PIC. (Esta batería sirve también para mantener viva la NVRAM adicional de que disponemos). El mismo DS1307 se encarga de realizar la conmutación entre una y otra por lo que no tenemos que tener en cuenta esta circunstancia y podemos olvidarnos de ella (salvo la de cambiar la pila cuando se agote).

3º El DS1307 tiene un pin de salida que, debidamente habilitado, nos ofrece una onda cuadrada con las frecuencias que puedes ver en la tabla superior. Esta salida es a colector abierto, por lo que es necesario, si la queremos utilizar para inyectarla en cualquier otro circuito, colocarle una resistencia pull-up de unos 10 Kohm a VCC.

Ten en cuenta que si nuestro DS1307 va a pasar grandes periodos de tiempo alimentándose solo de la batería el tener esta opción de salida habilitada consume cientos de veces más corriente que sin ella, por lo que podemos dejar la batería tiesa en muy poco tiempo. Si no es necesario es preferible deshabilitar esta opción (mas adelante veremos cómo hacerlo).

4º En la tabla superior podéis ver la estructura de la NVRAM, donde se mezclan tanto los registros de configuración, como los de salvaguarda de la fecha y hora del dispositivo, como asimismo los bancos de RAM de libre disposición para el usuario.

En esta tabla tener en cuenta que el Bit 7 de la dirección 0x00 hay que colocarla a 0 para que todo funcione. Es el Enable (habilitación) general del dispositivo.

5º El byte alojado en la dirección 0x07

Frecuencias de salida del DS1307 (Dallas Maxim)

RS1	RS0	SQUARE-WAVE OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

Una batería de Litio v a a permitir que el reloj siga su normal funcionamiento aún cuando apaguemos el PIC.

Estructura de la memoria NVRAM interna.

Timekeeper Registers

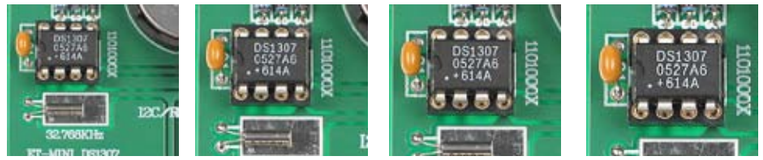
ADDRESS	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	FUNCTION	RANGE
00H	CH	10 Seconds			Seconds				Seconds	00–59
01H	0	10 Minutes			Minutes				Minutes	00–59
02H	0	12	10 Hour	10 Hour	Hours				Hours	1–12 +AM/PM 00–23
		24	PM/AM							
03H	0	0	0	0	0	DAY			Day	01–07
04H	0	0	10 Date		Date				Date	01–31
05H	0	0	0	10 Month	Month				Month	01–12
06H	10 Year				Year				Year	00–99
07H	OUT	0	0	SQWE	0	0	RS1	RS0	Control	—
08H-3FH									RAM 56 x 8	00H–FFH

0 = Always reads back as 0.

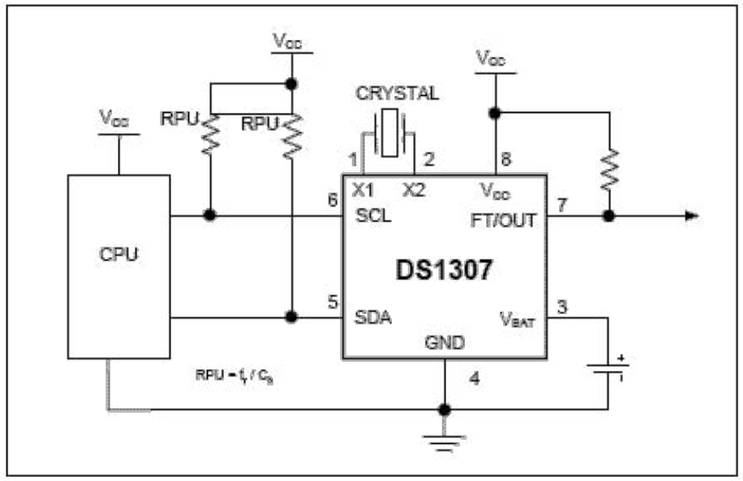
Registro de control.

CONTROL REGISTER

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0



TYPICAL OPERATING CIRCUIT



Circuito eléctrico sugerido por el fabricante.

El Relojito posee un DS1307, y con este articulo puedes aprender a sacarle provecho.

es el Control Register (registro de control) que nos permite configurar la función del pin de salida según los siguientes condicionantes:

El bit 4, SQWE, habilita o deshabilita la función de salida externa del pin Out.

El bit 7, OUT, establece el estado del pin de salida cuando SQWE está deshabilitado. Si OUT es 1 y SQWE es 0 entonces el pin de salida está en alto indefinidamente, si OUT es 0 y SQWE es 0 entonces el pin de salida está por el contrario en bajo indefinidamente.

Los bits 0 y 1 sirven para seleccionar la frecuencia de salida cuando SQWE está en alto según la tabla expuesta en 2º.

.Esquema eléctrico

Este el circuito propuesto por el fabricante como típico para su buen funcionamiento. A fines de explicar el uso de este integrado, vamos a replicarlo exactamente hasta el último detalle. Montaremos una placa que incluya todos los componentes necesarios, y que disponga de un conector compatible con mis placas de la serie RRBOARD2 (más datos sobre esta entrenadora en http://picmania.garciacuervo.com/Proyectos_**RRBOARD2.**htm)

Del pinout nada que añadir: (Ver Pin configurations)

Y este es el esquema definitivo que vamos a construir, en el que podemos destacar algunos detalles:

1º La comunicación con la RRBOARD2 la realizamos mediante nuestro buen amigo el conector CON-ML10 para cable plano de 10 hilos (alimentación y un puerto completo de 8 bits)

2º JP1-PB y JP3-PC permiten seleccionar la conexión de los SDA y SCL del I2C a los pines 0..1 ó 3..4 del puerto al que estén conectados (debido a que las familias 18F4550 y 16F877 implementa, el I2C en los pines RB0 / RB1 y en los RC3 / RC4)

3º JP2-OUT Permite conectar o desconectar el pin Out del DS1307 al pin Rx3 del puerto de la RRBOARD2 (Muy útil para usarlo con la Interrupción Externa 2 del 18F4550)

4º Dependiendo de dónde conectemos nuestro circuito en la RRBOARD2 podemos necesitar o no las resistencias Pull-Up imprescindibles para el bus I2C. Si lo conectamos al PORTB tenemos disponibles la internas del PIC, en cualquier otro caso podemos hacer uso del jumper JP3-PU para conectar dichas resistencias Pull-Up a VCC.

5º Añadimos además el jumper OUT para poder utilizar externamente la señal de onda cuadrada del pin Out del DS1307.

.Software:

Primero y antes que nada la **librería Driver CCS** C para el **DS1307**. Como comenté mas arriba esta librería es un gazpacho entre las varias que he encontrado por esos mundos de Dios, fundamentalmente las expuestas en la bibliografía que figura al final.

Las cosas que yo personalmente he introducido en este driver son:

- ***Parámetros** pasados a **ds1307_init()** para configurar en el inicio la función **OUT** del DS1307.*

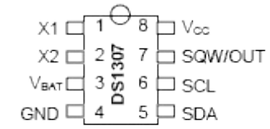
*-La funcionalidad **disable_interrupts(global) / enable_interrupts(global)** en cada una de las funciones definidas dependiendo del **#define use_interrups** en el programa principal.*

- *Función **ds1307_get_day_of_week()** que me devuelve el **string** con el **nombre del día** de la semana en la fecha actual (totalmente nueva y que no he encontrado por ahí).*

*- He añadido las funciones necesarias para escribir y leer todos los registros del **DS1307 ds1307_read_nvram_byte() y ds1307_write_nvram_byte()** ■*

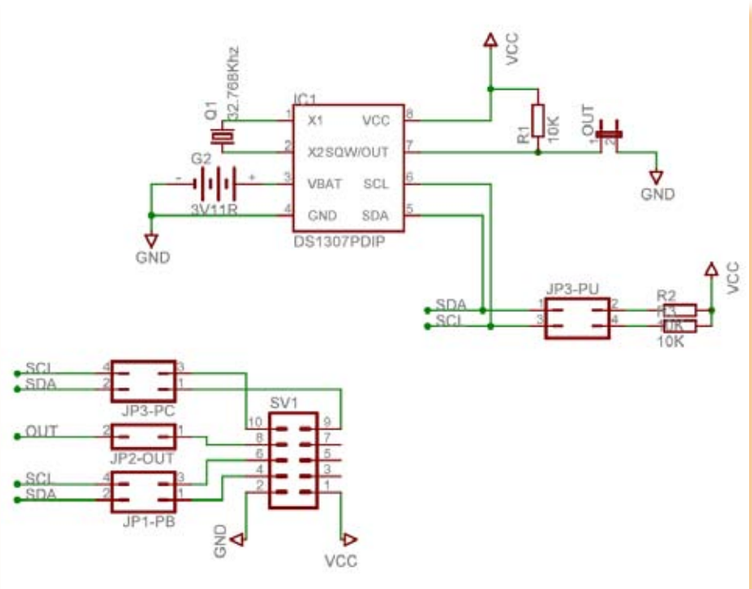
PIN CONFIGURATIONS

TOP VIEW



PDIP (300 mils)

Disposición de pines del DS1307.



Esquema propuesto para analizar el funcionamiento del chip.

```
////////////////////////////////////////////////////////////////////////////
///                            DS1307.C                                  ///
///                    Driver for Real Time Clock                        ///
///                    modified by Redpic 08/2006                        ///
///                 http://picmania.garcia-cuervo.com                    ///
///                                                                      ///
/// void ds1307_init(val)                                                ///
///                  - Enable oscillator without clearing the seconds register   ///
///                    used when PIC loses power and DS1307 run from 3V BAT      ///
///                  - Config Control Register with next parameters:             ///
///                     DS1307_ALL_DISABLED          All disabled                ///
///                     DS1307_OUT_ON_DISABLED_HIHG  Out to Hight on Disable Out ///
///                     DS1307_OUT_ENABLED           Out Enabled                 ///
///                     DS1307_OUT_1_HZ              Freq. Out to 1 Hz           ///
///                     DS1307_OUT_4_KHZ             Freq. Out to 4.096 Khz      ///
///                     DS1307_OUT_8_KHZ             Freq. Out to 8.192 Khz      ///
///                     DS1307_OUT_32_KHZ            Freq. Out to 32.768 Khz     ///
///                                                                      ///
///                     Example init:                                    ///
///                     ds1307_init(DS1307_ALL_DISABLED);                ///
///                     ds1307_init(DS1307_OUT_ENABLED | DS1307_OUT_1_HZ);       ///
///                                                                      ///
/// void ds1307_set_date_time(day,mth,year,dow,hour,min,sec) - Set the date/time  ///
///                                                                      ///
/// void ds1307_get_date(day,mth,year,dow)                   - Get the date       ///
///                                                                      ///
/// void ds1307_get_time(hr,min,sec)                         - Get the time       ///
///                                                                      ///
/// char ds1307_read_nvram_byte(char addr)                   - Read byte in address  ///
///                                                                      ///
/// void ds1307_write_nvram_byte(char addr, char value)      - Write byte in address ///
///                                                                      ///
/// void ds1307_get_day_of_week(char* ptr)                   - Get string Day Of Week///
///                                                                      ///
/// If defined USE_INTERRUPTS all functions disable Global Interrupts on starts and  ///
///                           enable Global on ends else usar can do it hiself       ///
///                                                                      ///
////////////////////////////////////////////////////////////////////////////

#ifndef RTC_SDA
#define RTC_SDA  PIN_B0
#define RTC_SCL  PIN_B1
#endif

#use i2c(master, sda=RTC_SDA, scl=RTC_SCL)

#define DS1307_ALL_DISABLED         0b00000000 // All disabled
#define DS1307_OUT_ON_DISABLED_HIHG 0b10000000 // Out to Hight on Disable Out
#define DS1307_OUT_ENABLED          0b00010000 // Out Enabled
#define DS1307_OUT_1_HZ             0b00000000 // Freq. Out to 1 Hz
#define DS1307_OUT_4_KHZ            0b00000001 // Freq. Out to 4.096 Khz
#define DS1307_OUT_8_KHZ            0b00000010 // Freq. Out to 8.192 Khz
#define DS1307_OUT_32_KHZ           0b00000011 // Freq. Out to 32.768 Khz

#define Start_user_address_nvram    0x08
#define End_user_address_nvram      0x3f

char days_of_week[7][11]={"Lunes\0","Martes\0","Miércoles\0","Jueves\0","Viernes\0","Sábado\0","Domingo\0"};

byte ds1307_bin2bcd(byte binary_value);
byte ds1307_bcd2bin(byte bcd_value);

void ds1307_init(int val){

   byte seconds = 0;

#ifndef USE_INTERRUPTS
   disable_interrupts(global);
#endif

   i2c_start();
   i2c_write(0xD0);
   i2c_write(0x00);
   i2c_start();
   i2c_write(0xD1);
   seconds = ds1307_bcd2bin(i2c_read(0));
   i2c_stop();
   seconds &= 0x7F;

   delay_us(3);

   i2c_start();
   i2c_write(0xD0);
   i2c_write(0x00);
   i2c_write(ds1307_bin2bcd(seconds));
   i2c_start();
   i2c_write(0xD0);
   i2c_write(0x07);
   i2c_write(val);
   i2c_stop();

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}

void ds1307_set_date_time(byte day, byte mth, byte year, byte dow, byte hr, byte min, byte sec){

#ifndef USE_INTERRUPTS
   disable_interrupts(global);
#endif

  sec &= 0x7F;
  hr &= 0x3F;

  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x00);
  i2c_write(ds1307_bin2bcd(sec));
  i2c_write(ds1307_bin2bcd(min));
  i2c_write(ds1307_bin2bcd(hr));
  i2c_write(ds1307_bin2bcd(dow));
  i2c_write(ds1307_bin2bcd(day));
  i2c_write(ds1307_bin2bcd(mth));
  i2c_write(ds1307_bin2bcd(year));
  i2c_stop();

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}

void ds1307_get_date(byte &day, byte &mth, byte &year, byte &dow){

#ifndef USE_INTERRUPTS
   disable_interrupts(global);
#endif

  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x03);
  i2c_start();
  i2c_write(0xD1);
  dow  = ds1307_bcd2bin(i2c_read() & 0x7f);
  day  = ds1307_bcd2bin(i2c_read() & 0x3f);
  mth  = ds1307_bcd2bin(i2c_read() & 0x1f);
  year = ds1307_bcd2bin(i2c_read(0));
  i2c_stop();

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}

void ds1307_get_time(byte &hr, byte &min, byte &sec){

#ifndef USE_INTERRUPTS
   disable_interrupts(global);
#endif

  i2c_start();
  i2c_write(0xD0);
  i2c_write(0x00);
  i2c_start();
  i2c_write(0xD1);
  sec = ds1307_bcd2bin(i2c_read() & 0x7f);
  min = ds1307_bcd2bin(i2c_read() & 0x7f);
  hr  = ds1307_bcd2bin(i2c_read(0) & 0x3f);
  i2c_stop();

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}


char ds1307_read_nvram_byte(char addr){

   char retval;

#ifndef USE_INTERRUPTS
```

```
   disable_interrupts(global);
#endif

   i2c_start();
   i2c_write(0xD0);
   i2c_write(addr);

   i2c_start();
   i2c_write(0xD1);
   retval = i2c_read(0);
   i2c_stop();

   return(retval);

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}

void ds1307_write_nvram_byte(char addr, char value){

#ifndef USE_INTERRUPTS
   disable_interrupts(global);
#endif

   i2c_start();
   i2c_write(0xD0);
   i2c_write(addr);
   i2c_write(value);
   i2c_stop();

#ifndef USE_INTERRUPTS
   enable_interrupts(global);
#endif

}

void ds1307_get_day_of_week(char* ptr){

   byte lday;
   byte lmonth;
   byte lyr;
   byte ldow;
   ds1307_get_date(lday,lmonth,lyr,ldow);
   sprintf(ptr,"%s",days_of_week[ldow]);
}

//////////////////////////////////////////////////////

byte ds1307_bin2bcd(byte binary_value){

  byte temp;
  byte retval;

  temp = binary_value;
  retval = 0;
  while(1){
    if(temp >= 10){
      temp -= 10;
      retval += 0x10;
    }else{
      retval += temp;
      break;
    }
  }
  return(retval);
}

byte ds1307_bcd2bin(byte bcd_value){

  byte temp;

  temp = bcd_value;
  temp >>= 1;
  temp &= 0x78;
  return(temp + (temp >> 2) + (bcd_value & 0x0f));
}

//////////////////////////////////////////////////////
```

En uControl encontrarás un programa de ejemplo que hace uso de esta librería, que puedes descargar y probar a gusto.

Bibliografía:

- El buscador del Foro Todopic (www.todopic.com.ar/foros) poniendo "DS1307" en la casilla de búsqueda oportuna. (Algunos cientos de referencias pacientemente leídas y digeridas).

- El artículo Another DS1307 Driver publicado en el Foro oficial de CCS C, cuya paternidad corresponde a un tal Phil, basado según dice él mismo en el trabajo de un tal Tullos.

- El articulo DS1307 RTC and 24LC256 EEPROM on the same i2c bus problem publicado en el mismo foro que el anterior pero esta vez realizado por unos tales Birumher y Highwind en un mano a mano digno de un buen par de toreros.

- Y por último el artículo DS1307 routines que para no ser menos también procede del Foro CCS C y de donde he sacado la idea de habilitar y/o deshabilitar las interrupciones.

Todo esto adobado por supuesto con San Google Bendito donde escribiendo "PIC" y "DS1307" aparece más información de la que estoy dispuesto a digerir.

Comunicación inalámbrica entre PICs

En este proyecto se puede aprender de forma sencilla como establecer una comunicación inalámbrica entre dos micros, enviando un cuarteto de bits, que posteriormente se puede ver en el micro receptor. La implementación del sistema esta desarrollado con un par de micros 16F628A, y dos módulos de radio.

//por:Juan Ricardo Clavijo Mendoza//
jrclavijo@hotmail.com

Para evitar los errores en los datos de llegada, es necesario implementar algún método que garantice la veracidad de la información.

Para algunos de los proyectos de control se hace necesaria la comunicación entre microcontroladores de forma inalámbrica. Esta es una tarea relativamente sencilla si se hace con las herramientas adecuadas, tales como un buen par de radios, un buen compilador de micros, y un mejor algoritmo para el tráfico de la información.

.Método

La clave fundamental de este proyecto esta en el medio de transmisión que se utilice. En el comercio se pueden conseguir una gran gama de radios de trasmisión y recepción, con diferentes características como: costo, alcance, formas de modulación, y complejidad en el manejo entre otras.

Para este proyecto trabajaremos con un par de radios muy sencillos de la compañía canadiense LAIPAC que produce diferentes soluciones en el campo de comunicaciones inalámbricas. Se trata de un par de radios de los cuales uno es transmisor y el otro es receptor.

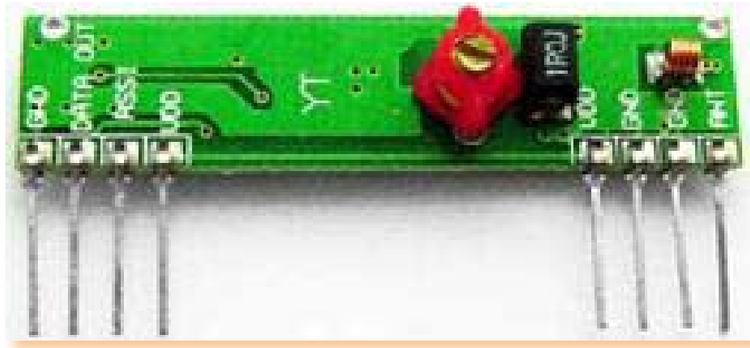
Las referencias son TLP434A y RLP434A, que son el transmisor y el receptor respectivamente. Este juego de radios trabaja una señal portadora de 434MHz y modulan en ASK, de tal manera que pueden transmitir valores lógicos 1 y 0.

La modulación ASK es similar a la modulación AM de la radio comercial de la banda de AM. En la modulación ASK un 0 lógico se representa con la ausencia de la señal portadora y un 1 lógico con la presencia de esta.

Los módulos de radio que se utilizan en este proyecto tienen un alcance de 100 metros si tienen una adecuada instalación de las antenas. La modulación ASK al igual que la modulación AM es supremamente propensa a las interferencias y al ruido. Por esta razón es importante implementar dentro de la programación del PIC una rutina que permita detectar cuando un dato ha llegado con errores para que sea descartado, ya que los módulos de radio no incluyen ningún método de software ni hardware para la detección de estos errores.

Los módulos de radio tienen la capacidad de transmitir a una velocidad de 9600 bits por segundo y de recibir a una velocidad de 4800 bits por segundo pero estos son los casos extremos de los módulos. Para establecer una comunicación más confiable, trabajaremos a una velocidad de 2400 bits por segundo.

Módulo trasmisor TLP434A."



La instalación de estos módulos de radio es muy simple, se utiliza dos pines para alimentar el modulo, uno con Vcc y otro con GND, un pin para la antena y otro para la entrada o salida de datos de forma serial.

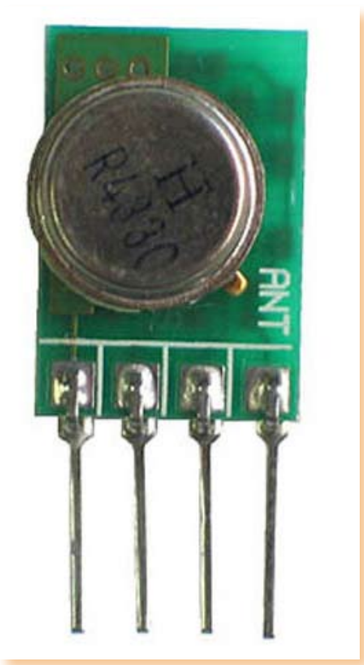
El modulo transmisor se puede alimentar con una tensión de entre 3V y 12V. La potencia de transmisión será mayor a mayor voltaje. El modulo receptor solo se puede alimentar con 5V.

.Implementación

Para establecer la comunicación entre los dos PIC trabajamos con el modulo USART de cada uno de los micros. Esto implica que uno de ellos será el transmisor y el otro el receptor. Estos micros los llamaremos, de aquí en adelante, TX y RX. En el micro TX se evalúan constantemente cuatro pines, en los cuales están instalados sendos pulsadores que son los cuatro bits de información que deseamos transmitir. Esta información es empaquetada y transmitida serialmente por la USART.

Para evitar los errores en los datos de llegada, es necesario implementar algún método que garantice la veracidad de la información. Para esto existen formas complejas de control de errores pero para este proyecto implementaremos un método muy sencillo conocido como redundancia, el cual consiste en transmitir repetidamente el mismo dato y verificar si el dato que llega en el micro RX es igual, para determinar que el dato no tiene errores.

Apariencia del receptor RLP434A.



En el micro RX están instalados cuatro LEDs que permiten ver el dato que llega de manera inalámbrica.

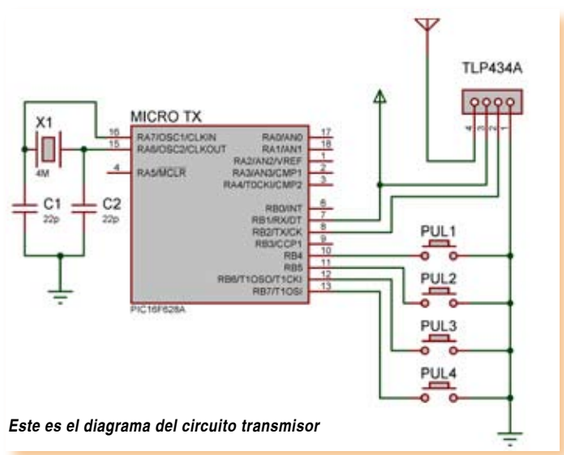
A continuación puedes ver el código fuente en C de los micros TX y RX. El compilador utilizado es el PICC de Hi-Tech, pero puede ser emigrado a cualquier otro compilador que trabaje en C.

.Programa del MICRO TX

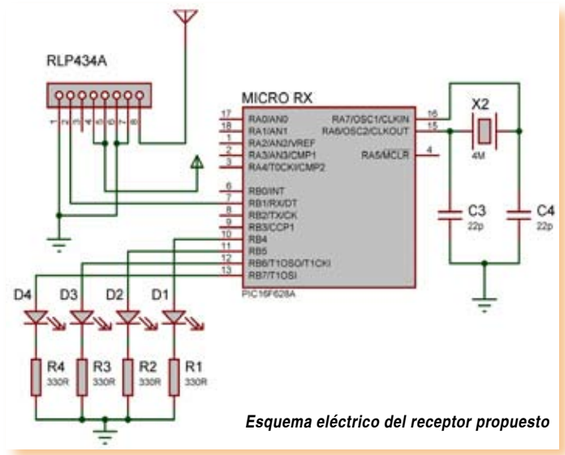
```
#include <pic.h>

void TxSerial( char d_ ) // Función para transmitir un dato de forma serial
{
         TXREG = d_;
         while( !TRMT );
}

void InicioSerial( void ) // Función d inicio de la USART
{
         TRISB1 = 1;
         TRISB2 = 0;
         TXSTA = 0x24;
         RCSTA = 0x90;
         BRGH = 1;      // Configuración de la USART a 2400 bits por segundo
         SPBRG = 103;
}

void main( void ) // Funcion principal
{
        char DATO;
        INTCON = 0;    // Configuración de las interrupciones
        TRISB = 0xFF; // Se configuran los pines de los leds como salidas
        RBPU = 0; // Se activan las resistencias PULL-UP
        InicioSerial(); // Función d inicio de la USART
        while(1)
        {
       DATO = (~PORTB>>4)&15; // Se guarda en la variable DATO el valor de los 4  pulsadores
       TxSerial( 170 ); // Se transmite una bandera de inicio con la secuencia de bits: 10101010
       TxSerial( DATO ); // Se transmite el dato de manera redundante. 8 veces
       TxSerial( DATO );
       TxSerial( DATO );
       TxSerial( DATO );
       TxSerial( DATO );
       TxSerial( DATO );
       TxSerial( DATO );
        }
}

 __CONFIG( 0x3F09 );
```

.Programa del MICRO RX

```
#include <pic.h>

// Declaración de variables de trabajo
char n=0;
char Trama[4]={1,2,3,4};
char DATO;

void InicioSerial( void )// Función para la configuración de la USART.
{
         TRISB1 = 1;
         TRISB2 = 0;
         TXSTA = 0x24;
         RCSTA = 0x90;
         BRGH = 1;
         SPBRG = 103;
}

void interrupt VET( void ) // Vector de interrupciones.
{
     if( RCIF ) //Interrupción serial
     {
                  DATO = RCREG; //Lectura del buffer de entrada serial
                  switch( DATO ) // Se evalúa el dato que llega
                  {
case 170:  for( n=0; n<4; n++ )Trama[n]=n; n=0; break; // Caso de la bandera de entrada.

                           default :  Trama[n++]=DATO; // Se guardan los datos de entrada en el búfer de la trama.
                                   if( n==4 ) // Se evalua cuanda a llega el cuarto byte de la trama.
                                    {
                                        if( Trama[1]==Trama[2] ) // Se comparan los datos 1,2,3 del bufer de la trama
                                                                 if( Trama[2]==Trama[3] ) // y se verifica que sean iguales.
                                                                 {
                                                                  PORTB = Trama[1]*16; // cuando el dato es correcto se muestra por
                                             for( n=0; n<8; n++ )Trama[n]=n;// el puerto b en los LEDs
                                                                 }
                                        n=4;
                                                              }
         }
         RCIF=0;
     }
}

void main( void ) // Funcion principal.
{
        INTCON = 0; // Se apagan todas las interrupciones.
        PEIE=1; // Se activan las interrupciones periféricas.
        RCIE=1; RCIF=0; // Se activan las interrupciones por recepción serial.
        GIE = 1; // Se activan las interrupciones de forma general.
        TRISB = 0x0F; // Se configuran los pines b como entrada y salida.
        PORTB = 0; // Se apagan los pines del puerto b
        InicioSerial(); // Se inicializa la USART.
        while(1); // bucle infinito para la espera de interrupciones.
}

__CONFIG( 0x3F09 );
```



Este es el diagrama del circuito transmisor



Esquema eléctrico del receptor propuesto

Paleotrónica
Ordenadores Sinclair

En este proyecto se puede aprender de forma sencilla como establecer una comunicación inalámbrica entre dos micros, enviando un cuarteto de bits, que posteriormente se puede ver en el micro receptor. La implementación del sistema esta desarrollado con un par de micros 16F628A, y dos módulos de radio.

//por:Ariel Palazzesi//
arielpalazzesi@gmail.com

Hijo de George William Carter Sinclair y Thora Edith Ella Sinclair, ambos ingenieros,

Clive Marles Sinclair nació el 30 de Julio de 1940, en Inglaterra.

Desde muy joven mostró gran interés por las matemáticas y la electrónica, investigando todo lo relacionado con diseños electrónicos y circuitos de radios. En Julio de 1961, fruto de este interés, Clive Sinclair funda Sinclair Radionics LTD , que sólo un año más tarde lanzaba su primer producto: un kit de un amplificador de audio que se vendía por correo. En los años siguientes la empresa comercializaría productos innovadores, como la primera calculadora de bolsillo (la Sinclair Executive, de 1972) o la primera TV en miniatura (la Microvision TV1A, comercializada en 1977 pero concebida casi 10 años antes).

Pero pese a los éxitos iniciales, a finales de los años 70, Sinclair enfrentaba una crisis financiera importante, que luego de varios cambios, fusiones y modificaciones en su empresa, lo lleva a fundar la ya mítica Sinclair Research LTD, en marzo de 1981.

Sería Sinclair Research LTD la responsable del lanzamiento de una gama de productos informáticos que harían historia, entre los que se incluyen el ZX80 (de 1980), el ZX81 (de 1981), el ZX Spectrum (de 1982), el Sinclair QL (de 1984), etc. Tal fue el éxito de la empresa y sus aportaciones al sector tecnológico que en 1983 la reina Isabel II otorgó a Clive Sinclair el título de "Sir", como reconocimiento por "sus servicios a la industria británica".

Estos ordenadores serian los que harían famoso el nombre de este inglés que nunca se recibió de ingeniero. A continuación, un breve resumen de cada uno de ellos.

> Estos ordenadores serian los que harían famoso el nombre de este inglés que nunca se recibió de ingeniero.

. ZX80

Comercializado a partir de 1980, fue el primer ordenador del Reino Unido que pudo comprarse por menos de libras (costaba £99.95). Se entregaba como un kit, y los compradores tenían que ensamblarlo y soldar las piezas. Pero por un precio ligeramente más alto podía obtenerse una versión lista para usar. El ZX80 fue muy popular en su época, llegando a existir una lista de espera de varios meses para poder comprarlo.

Su hardware se basaba en un microprocesador NEC μPD780C-1 (clon del Zilog Z80), que corría a 3.25 MHz. Disponía de 1 KB de memoria RAM estática, ampliable a 16K; y 4 KB ROM, que contenían el lenguaje de programación Sinclair BASIC, el editor, y el Sistema Operativo. Los comandos de BASIC, al igual que en los ordenadores que Sinclair inventará más tarde, no se escribían. En lugar de ello, cada tecla tenía diferentes funciones, accediendo a cada una de ellas presionando teclas especiales de cambio.

Utilizaba un televisor a modo de monitor, y el almacenamiento de datos y programas se hacía en cinta de casete. El generador de video del ZX80 era muy primitivo, resolviéndose la mayoría de las tareas implicadas en la generación de la señal de video mediante software. Esto hacia que el ordenador solo pudiese generar imágenes cuando estaba "desocupado", es decir, cuando estaba esperando a que el usuario presionase una tecla. Cuando estaba corriendo un programa BASIC, la pantalla se ennegrecía. No era posible mostrar gráficos u objetos animados, y las imágenes eran en blanco y negro.

ZX Modelos:

Sinclair ZX80
Año:1980
Micropro NEC μPD780C-1 de 3.25 MHz
1 KB RAM (ampliable a 16 KB)
4 KB ROM

La memoria del ZX80 podía ampliarse a 3KB mediante un "RamPac". La máquina se alojaba en una pequeña caja plástica de color blanco, que disponía de un teclado de membrana de color azul en el frente.

Este ordenador inició la locura por los "Home Computers", en los 1980s. Las ventas del ZX80 superaron las 50.000 unidades, colocando a Inglaterra en una situación de liderazgo en el campo de los ordenadores domésticos. Debido a su nada sofisticado diseño y su tendencia al sobrecalentamiento, son pocas las máquinas sobrevivientes en buen estado de funcionamiento, por lo que alcanzan altos precios entre los coleccionistas.

.ZX81

Un año después de revolucionar la industria con el ZX80, Sinclair lanzo el ZX81. Se trataba de una versión mejorada del anterior, montado en una caja negra con un teclado también de membrana. El video seguía desplegándose sobre un televisor, pero esta vez el hardware era algo mas sofisticado, y gracias a un modulador de radiofrecuencia ya no se perdía la imagen mientras el ordenador "trabajaba".

La pantalla mostraba texto solamente, con 32 caracteres de ancho por 24 de alto. Sin embargo, eran posibles mostrar gráficos con una resolución de 64 por 48 píxeles mediante el uso del comando PLOT, que ingeniosamente seleccionaba uno de los 16 caracteres gráficos disponibles, de 2x2 pixeles.

El dispositivo de almacenamiento era el mismo que en el ZX80: un grabador de casetes y cintas magnéticas de audio. Su microprocesador, un NEC compatible con el Zilog Z80, tenía un ciclo de reloj de 3.5 MHz. Lo acompañaban solamente tres chips: un ASIC a medida (ULA, de Ferranti); una RAM del tipo 4118, de 1Kx8 bits; y un chip de memoria ROM tipo 2364, de 8Kx8 bits. El BASIC soportaba aritmética de coma flotante.

> En 1985 vio la luz el Spectrum 128, que era básicamente un Spectrum con más memoria y una carcaza "de verdad".

Podías comprar este ordenador, en el Reino unido, por £70 (unos u$s 100). Como la memoria de la máquina era muy escasa, se ofrecía una ampliación de 16KB de RAM (u$s 100). Más tarde, a mediado de 1982, se pusieron a la venta ampliaciones de 32KB y 64KB. Rápidamente se hicieron famosos por la poca calidad de sus contactos, que ante el menor movimiento o vibración se desconectaban y hacían perder los resultados de horas de programación.

El ZX81 tenia la capacidad de utilizar una impresora térmica, en la que un alambre caliente dibujaba los puntos sobre un papel térmico color gris de 4 pulgadas de ancho.

A pesar de sus limitaciones, había muchos juegos y aplicaciones que funcio-

Sinclair ZX81
Año:1981
Versión mejorada de la Sinclair ZX80. Su Hardware era más sofisticado.

ZZ Spectrum 128 y ZZ Spectrum
Año:1982
Ordenador de 8 bits basado en el Z80, funcionando a 3.5 MHz
RAM de 16KB o 48KB
ROM 16KB
Manejaba gráficos acolor.

A pesar de sus limitaciones, había muchos juegos y aplicaciones que funcionaban en la minúscula memoria de 1 K, incluyendo un juego básico de ajedrez.

naban en la minúscula memoria de 1 K, incluyendo un juego básico de ajedrez. Para los aficionados no resultaba difícil de conocer, entender, y controlar totalmente el ordenador, casi imposible de lograr con los ordenadores de hoy en día.

A los fanáticos de las comparaciones les gustará saber que un bucle FOR-NEXT, contando de 1 a 1.000, tardaba 19 segundos en ejecutarse. Otra curiosidad era que, si bien no disponía de la posibilidad de generar sonido, algunos programadores modulaban la interferencia que el procesador causaba en la TV, creando algunos sonidos rudimentarios. También había un notorio bug que hacía que (algunos) ZX81s calcularan la raíz cuadrada de 0.25 como 1.3591409 en lugar de 0.5.

.ZX Spectrum

El 23 de abril de 1982 Sinclair lanzó el ordenador que se convertiría en su mayor éxito, y que hoy es un objeto de culto: el ZX Spectrum.

Al igual que sus predecesores fue un ordenador de 8 bits basado en el Z80, funcionando a 3.5 MHz. Lo primero que asombraba de esta máquina era la posibilidad de utilizar gráficos en colores.

Sinclair proporcionaba este ordenador con dos configuraciones de RAM diferentes: 16 KB o 48 KB. Si le sumamos los 16K de ROM, la memoria total de ambos modelos era realmente de 32KB y 64 KB, al límite del direccionamiento posible con 16 bits.

El teclado de membrana utilizado en los modelos anteriores se reemplazó por uno de caucho que, sin ser la panacea, facilitaba su uso.

El sistema de almacenamiento en cinta (casete de audio común) funcionaba a 1.200 baudios, y un juego de 48 KB tardaba unos 5 minutos en cargar. Con "turbo", nombre elegido por algunos programadores para denominar rutinas propietarias más rápidas, podía reducirse en algo este tiempo. Si querías más velocidad (pero no mucha), podías comprar una unidad de disco muy primitiva, llamada microdrive, muy popular entre los desarrolladores de software.

Una de las particularidades del ZX Spectrum fue su sistema de vídeo, capaz de mostrar 256x192 píxeles con 15 colores en menos de 8 KB. Esto era posible gracias a que resolución de estos colores es a nivel de carácter (8x8 píxeles). El problema de tener distintas resoluciones para la luminosidad y el color volvía locos a los programadores de juegos, que debían minimizar las colisiones entre colores. Este fenómeno era conocido como "attribute clash" en el mundo anglosajón.

En 1985 vio la luz el Spectrum 128, que era básicamente un Spectrum con más memoria y una carcaza "de verdad". Las teclas finalmente se parecían a las de un ordenador actual, y la cantidad de memoria disponible hacia innecesarios los módulos de ampliación.

A medida que la tecnología abaratava los costos, Sinclair diseñó y comercializó otros modelos, llamados ZX SPECTRUM 128 +2 y ZX SPECTRUM 128 +3, que dentro de la carcaza incorporaban la unidad de cinta o de microdrives, respectivamente.

Su optimizado y compacto diseño hizo las delicias de miles de aficionados a la informática y los videojuegos.

Arkanoid en el ZX Spectrum de 48KB

Placa principal del ZX81

Diseño y Diagramación
azimut.estudio@gmail.com / la plata / bs as / argentina