[Analógico • Digital • Microcontroladores & Embebido • Audio • Test & Medida]

Diciembre 2011 N.º 378 6,50 €

elektor

www.elektor.es

Tableta Android
Control de disparo fotográfico

Código de bloqueo a toda prueba
Con encriptación AES de 128 bits

Pendrive USB en un micro
Almacenamiento conveniente y seguro

Vela LED Electrónica
¡Puedes apagarla soplando!

ISSN 0211-397X

La toma de aire de Elektor

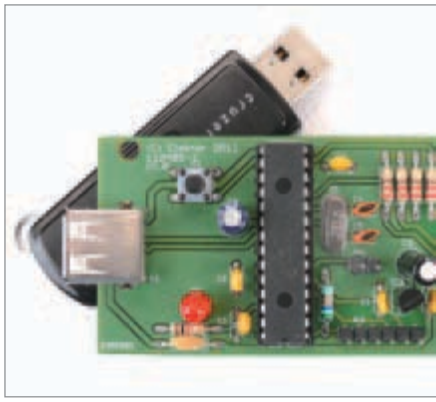
Una de las preguntas que más frecuentemente me hacen, normalmente por email aunque ocasionalmente por teléfono o correo ordinario (¡si!), es: “puedo colaborar con su maravillosa publicación y si es así ¿cuáles son los requisitos y los temas específicos que les interesan?” Invariablemente la respuesta es: “Si, por favor, estamos deseando evaluar la posibilidad de publicar tu propuesta de artículo. Cada edición de Elektor incluye una amplia variedad de temas y campos de interés. Por favor, consulta nuestra Guía para Autores que encontrarás en www.elektor.es en la pestaña Servicios”. Aquí suelo añadir algunas palabras de aliento como “estamos haciendo esto en España desde 1980” y algunas indicaciones sobre el tamaño de los artículos. Algunos llevan muchos años colaborando con nosotros y saben perfectamente cómo hay que entregar el material y las preferencias sobre las imágenes que ilustrarán el artículo; otros necesitan de más ayuda con los formatos de los archivos, cuestiones gramaticales, el estilo o profundidad del contenido técnico que pueden manejar.

La toma de aire es en realidad un colector – las colaboraciones de empresas, periodistas y especialistas de la industria también son bienvenidas. Este enfoque funciona bien, pero también crea una cola de artículos esperando ser publicados, y en muchos casos hemos de confirmar a los autores que no se han perdido y que en el laboratorio se está trabajando en su proyecto.

Ahora, un poco de solemnidad: por desgracia, aproximadamente 7 de cada 10 propuestas que nos llegan a través de todos los canales internacionales, son rechazadas. Las razones por las que el equipo de editores y diseñadores es tan duro y cruel con los autores en ciernes son diversas: el uso inadecuado de componentes, lo mismo para los componentes obsoletos; refritos de las hojas de datos de los fabricantes o antiguos artículos de Elektor (!); circuitos nebulosos pillados de webs nebulosas, diseños electrónicos pobres e intentos de utilizar la revista para eliminar el stock que está cogiendo polvo en un almacén. El resto es felizmente considerado para su publicación o post-ingeniería en nuestro laboratorio, no importa si está pobremente escrito o el prototipo está montado en una palca de pruebas – en general tenemos buen humor con un buen ojo para la originalidad. Incluso si tardamos algún tiempo en contactar contigo debido a la carga de trabajo que tenemos en Elektor, danos una oportunidad y eventualmente veras tu nombre (¡y tu circuito!) impreso en la revista – de ninguna manera es difícil, estamos aquí para ayudar.

¡Disfruta de esta edición!
Eduardo Corral, Editor

elektor



¡Todos en Elektor os deseamos una muy Feliz Navidad!
Gracias por vuestro continuo apoyo.

SUMARIO

Volumen 32
Diciembre 2011
nº 378

20 Pendrive USB en un microcontrolador

El registrador de datos USB aquí descrito es una solución universal de bajo consumo al problema. Coge todos los datos en serie enviados desde un sistema microcontrolador externo y los almacena en un archivo en un pendrive USB que puede ser analizado posteriormente con un PC.

32 Time-lapse con una Tableta Android

El diseño aquí descrito se puede utilizar con una cámara fotográfica para hacer que esta tome fotografías a intervalos regulares. Si con esas imágenes haces una película, el resultado es lo que se llama una película “time-lapse” en la que horas, o incluso días, se reducen a unos cuantos segundos. El proyecto aquí descrito opera sobre el botón de disparo de la cámara mecánicamente, empleando para ello un servo de los utilizados en los modelos RC.

36 Android como entorno de desarrollo

Los ordenadores Tablet (PCs) con sistema operativo Android están disponibles ahora por debajo de los 100 €. Incluyen toda la electrónica y son visualmente atractivos. Muchas funciones cuya implementación en un entorno embebido supone un esfuerzo considerable son una prestación estándar en los ordenadores tablet. En este artículo examinamos los factores que intervienen en la utilización de tablets en proyectos electrónicos.

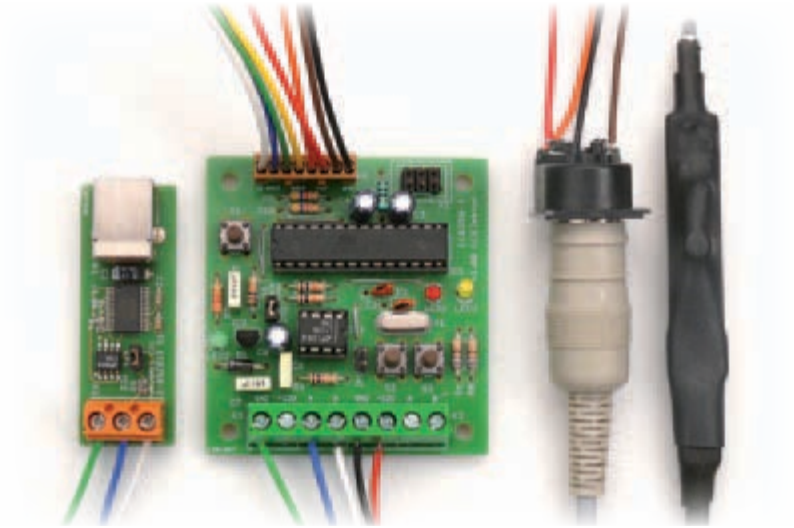
44 Vela LED electrónica

Podemos encontrar en el mercado imitaciones de velas que utilizan LEDs como elemento de iluminación. Pero aquí estamos describiendo un proyecto ligeramente distinto con algunas características poco usuales – ¡después de todo, las velas se apagan soplando!

¡Que viene el bus! (10)

Obteniendo valores de medida con 22 bits

Jens Nickel



Ahora se trata de alta resolución: en nuestro nodo de bus conectamos un ADC externo capaz de muestrear valores a 22 bits. Gracias al puerto SPI, la lectura de los datos en el microcontrolador resulta bastante fácil. También su representación en el PC se logra rápidamente; sólo tenemos que reajustar ligeramente la página HTML de la última entrega.

Lo bonito de la electrónica es que uno puede inspirarse fácilmente en otros proyectos. En este caso ocurrió con el proyecto de un lector llamado “medidor de gradiente de temperatura” [1] que encontrarás en esta misma edición. Para medir cambios en la temperatura con alta precisión, nuestro lector el Dr. Dietmar Schröder construyó un circuito de medida con un ADC externo (MCP3551 de Microchip),

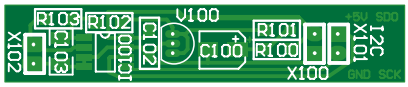


Figura 2. Tarjeta de Dietmar Schröder. X100 y X101 se conectan con K4 en el nodo experimental.

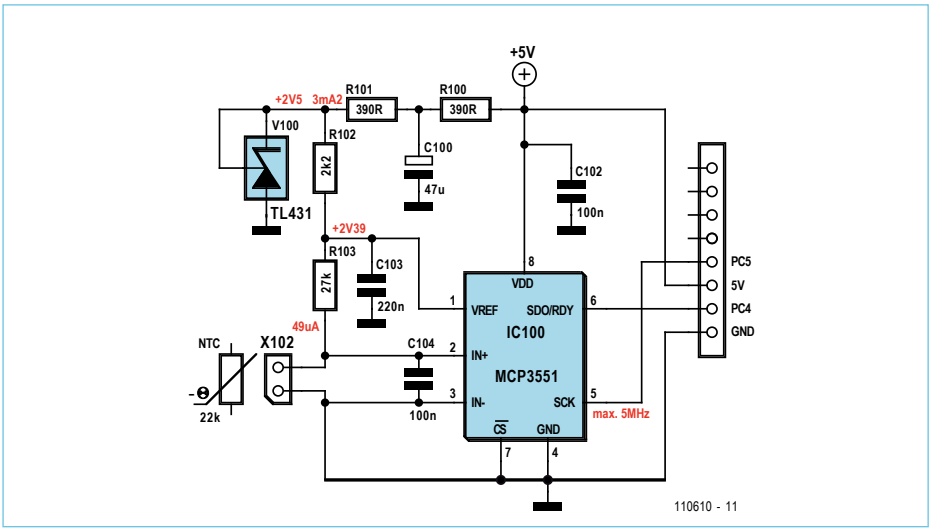


Figura 1. Esquema de circuito del sensor de temperatura de alta resolución, que incorpora una conexión para nuestros nodos experimentales.

Figura 3. Representación de los valores del ADC y de la temperatura en el ElektorBusBrowser.

Productos de Elektor y soporte

- Nodo experimental (Tarjeta 110258-1 o set de tres tarjetas 110258-1C3)
- Conversor USB/RS485 (montado y probado 110258-91)
- Descarga gratuita de software (firmware del controlador más el software de PC)
- Todos los productos y descargas están disponibles en la web de este artículo: www.elektor.de/110610

capaz de convertir valores de tensión con una resolución de 22 bits para los datos. La **figura 1** muestra el pequeño circuito con este conversor Delta-Sigma, la NTC y algunos componentes más. En la **figura 2** puede verse la correspondiente tarjeta del autor, que puede descargarse en [1]. El conversor A/D suministra en su salida digital –un puerto SPI– aproximadamente 14 valores de medida por segundo. Este circuito fue creado para ser conectado en un nodo experimental. A través de dos pines podemos leer los valores de medida en nuestro conector de ocho líneas [2], y mediante otros dos alimentamos el circuito del sensor. Y ya que el "medidor de de gradiente de temperatura" utiliza un ATmega para el tratamiento de los datos, presentamos además un buen código base (también descargable en [1]).

Bits rebotando

El puerto SPI se gestiona mediante "Bit-Banging", o sea, de forma manual por software y no mediante el hardware del propio puerto SPI del controlador. Pude exportar la función en C del código original fácilmente a BASCOM (**código 1**). En `Readexternaladc()` primero se espera hasta que el ADC ponga el pin SDO a nivel bajo. Esto indica que puede llamarse a un nuevo valor de medida. Después el controlador conmuta alternativamente la línea SCK entre nivel bajo y alto. Tras cada flanco creciente aparece un bit del valor de medida digitalizado en la línea SDO, transfiriéndose primero los bits más significativos. La variable long `Dat` sirve para la recepción del resultado. Una vez que SDO esté a nivel alto, se añade un bit=1 totalmente a la derecha del valor, y todos los bits de `Dat` se desplazan una posición hacia la izquierda. En la hoja de datos [3] podemos consultar por qué motivo han de leerse 24 bits y por qué se ha añadido un reloj adicional por seguridad. Lo importante para nosotros ahora es: el ADC en el circuito funciona en "Continuous Conversion Mode". No tenemos que hacer nada más, salvo darle tensión, y empezará a tomar medidas.

Como se describe en el artículo del proyecto original, considerando el ruido inevitable, sólo 19 bits son significativos (lo cual todavía nos deja una impresionante resolución de un nivel de 3/10000). Por ello, en el resultado podemos eliminar sin problema los dos bits inferiores, desplazando todo hacia la derecha. Nos quedarían 20 bits, siendo el primer bit en un rango de temperatura de hasta -35°C siempre 0. Esto ajusta perfectamente con nuestro *ApplicationProtocol*, ya que permite transferir enteros de hasta 19 bits (más el signo) [4]. En el código en BASCOM descargable [5] puede verse fácilmente cómo los 19 bits se dividen en tres bytes de datos para su transferencia.
No resultó difícil para nosotros llevar a cabo los primeros tests ya que el autor nos cedió temporalmente su tarjeta con el sensor. Sólo necesitaba un interfaz de usuario para el PC. Éste puede realizarse con el concepto presentado en la edición anterior. Se basa en páginas HTML que se representan en el *ElektorBusBrowser*. Sólo tuve que adaptar en la página HTML "Index.htm" de la última entrega de la serie que en lugar de un "VALUE2" sólo se recibe "VALUE4" y mostrarlo en el cuadro de texto.

Código 1: código en BASCOM para la lectura de valores del ADC

```
Function Readexternaladc() As Long

   Dat = 0
   Sck = 1
   Notimeout = 100

   While Sdo = 1 And Notimeout > 0
      Notimeout = Notimeout - 1
      Waitms 1
   Wend

   For Ia = 0 To 23
      Sck = 0
      Waitus 80

      Shift Dat , Left , 1

      Sck = 1
      If Sdo = 1 Then
         Dat = Dat + 1
      End If
      Waitus 80

   Next

   Sck = 0
   Waitus 80
   Sck = 1
   Waitus 80

   If Notimeout = 0 Then
      Readexternaladc = 0
   Else
      Readexternaladc = Dat
   End If

End Function
```

Interfaz de usuario

Uno de los nodos se convertirá en un medidor de gran resolución, incluso el movimiento de la mano alterará los valores del ADC. Naturalmente, también quería que los valores de temperatura se dieran en grados Celsius. En el firmware para los nodos sensores he implementado la correspondiente rutina de cálculo, tomando la curva característica de la NTC (en el 5° paso) en el código del "medidor de cambios de temperatura" y ajustándola en consecuencia. Mi demo del firmware no incluye cifras en coma flotante ni fracciones. Por lo tanto, el resultado son valores en 1/10000 o 1/1000 gra-

Código 2: script en la página HTML

```
var SetFlag = false;
var QuantityToSet = 0;
var ScaleToSet = 0;
var DisplayScale = 0;


function ProcessPart(part)
{

  if ((part.Sender == 2) && (part.Parttype == PARTTYPE_VALUE4))
  {
    if (part.Channel == 0) {TextboxSetvalueScaled('ADC', part.Numvalue, DisplayScale);};
  }

  if (SetFlag==true)
  {
    if ((part.Sender == 2) && (part.Parttype == PARTTYPE_SCALE) && (part.Channel == 0))
    {
      if (QuantityToSet==TEMPERATURE) {TextSetvalue('unit','°C');};
      if (QuantityToSet==RAWVALUE) {TextSetvalue('unit','ADC-Value');};
      DisplayScale = ScaleToSet;
      SetFlag = false;
    }

    var parts = InitParts();
    parts = SetScale(parts, 10, 2, 0, 0, QuantityToSet, 0, ScaleToSet);
    SendParts(parts, true);
  }
}


function SetSensorScaleIndirect(quantity, scale)
{
  SetFlag = true;
  QuantityToSet = quantity;
  ScaleToSet = scale;
}
```

dos, que pueden transferirse como enteros de 4 bytes (nota: sin calibración, la precisión absoluta de los valores de temperatura lamentablemente no será muy alta).
La selección (del ADC, para valores de 1/1000 o 1/10000) se hace lógicamente con un botón en el interfaz de usuario en HTML (ver la **figura 3**). Esto también nos sirve perfectamente para mostrar cómo se ajusta la magnitud física, la unidad y el escalado de un sensor. Los bytes de datos a enviar tienen la representación decimal *40, 193, 33, -4* para valores en 1/10000 grados (se utiliza el canal 0 del nodo).
Gracias a la librería Javascript JSBus no tendremos que calcular los bytes por nuestra cuenta, sencillamente incluimos el siguiente script en nuestra página HTML, que ya genera y envía las *parts*:

```
var parts = InitParts();
parts = SetScale(parts, 10, 2,
0, 0, TEMPERATURE, 0, -4);
SendParts(parts, true);
```

Transmisión fiable

Se llama a las líneas de código arriba citadas sólo tras pulsar el botón correspondiente en el formulario HTML, igual que se describió en la última parte de la serie sobre el bus [6]. Sin embargo, en las pruebas me percaté de que los clics no siempre conducen al resultado deseado. No obstante, el cuadro de texto *OutCommand* del ElektorBusBrowser sí que se calculaban correctamente los bytes de datos del mensaje. Probablemente se deba a un problema en el firmware del nodo. Mi deducción es que el mensaje del sensor no puede recibirse correctamente mientras está ocupado con la lectura de los ADCs externos. Resulta interesante ver cómo la transmisión de los comandos gana en fiabilidad si reducimos `intFreeBusTime` en el código fuente del ElektorBusBrowsers (explícitamente, en esta aplicación ya no necesitamos más *FreeBusPhases*, sin embargo, dejamos abierta la puerta a implementar esta posibilidad posterior-

mente en Javascript). Ya que todavía no disponemos de ningún monitor del bus ni algún método de depuración para el firmware, todavía no he sido capaz de dar con el error. En fin, sirva esto para demostrar al menos cómo podemos afrontar tales problemas.

El **código 2** muestra una solución. Los botones para seleccionar la unidad llaman a una rutina en Javascript nombrada `SetSensorScaleIndirect(…)`, la cual sin embargo no conduce al comando correspondiente, sino tan solo activa un flag. Mientras este flag esté activo, el master repetirá el envío del comando. Además, podemos colocar las líneas de código necesarias en la rutina `ProcessPart(part)`, la cual es llamada periódicamente por la librería JSBus en caso de llegar un valor de medida.

El flag se desactiva cuando se recibe una confirmación por parte del nodo (el firmware en BASCOM incluye los cuatro bytes necesarios, que se colocan junto con el valor de medida en los mensajes enviados regularmente). Sólo así se cambia la unidad en la página HTML. Para la representación decimal con coma del valor en el cuadro de texto, se utiliza la función `TextboxSetvalueScaled(…)` (implementada en la nueva versión de JSBus [5]). El archivo HTML descargable también incorpora algo de CSS [7], para dotar de mejor apariencia al interfaz de usuario.

Pequeña “optimización”

En algunos tests ocurría que la representación de valores se quedaba bloqueada durante algunos minutos. Gracias a uno de los osciloscopios del laboratorio pude comprobar que algunos bits todavía circulaban por el bus; sólo hasta que desactivé el planificador. Entonces hice que todos los bytes del bus se mostrasen en un cuadro de texto independiente, y vi que sólo se enviaban continuamente *FreeBusMessages* del planificador. Lo que estaba mal aquí no resultaba tan fácil de averiguar, considerando que había tres procesos intercambiando mensajes entre sí. Primero, el bucle del planificador, llamando a los nodos uno tras otro. Segundo, la rutina `ShowMessage`, que funciona en paralelo cuando se recibe un mensaje de 16 bytes; si este mensaje se recibe desde el nodo al que se ha llamado, se pasa al siguiente nodo en la serie. Y tercero, entra en juego un timer; cuando un dispositivo del bus no responde en un determinado tiempo, se pasa al siguiente nodo. Finalmente me di cuenta de que había cometido un gran error. Para indicar qué nodo era el próximo en la serie, se le permitía a todas las rutinas cambiar la variable global `intPolledNodesCursor`, que apuntaba al siguiente nodo en la lista del planificador. Pero cuando esta variable se cambiaba desde el exterior, mientras el bucle del planificador estaba en marcha, resultaba muy difícil depurar los efectos secundarios que podían aparecer. Cambié el código de modo que ahora sólo se utiliza un flag llamado `boolNextNode` para pasar al próximo nodo. Este flag se comprueba al inicio del bucle del planificador y no tiene ningún otro efecto sobre la ejecución del código. ¡Y ahora ya era estable! Afortunadamente los archivos para la instalación de la anterior parte de la serie todavía no se habían subido a la página web. En las descargas en [5] también se encuentra el ElektorBusBrowser mejorado, como código fuente VB.NET y archivo .exe.

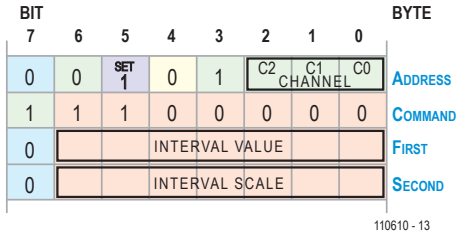
Algunas ideas

Me hubiera gustado integrar algunas características más, como la representación de cambios en la temperatura, incluyendo un filtrado con parámetros ajustables. Con ello hubiéramos podido encontrar un buen sustituto para la tarjeta del procesador, el display y los potenciómetros de ajuste del proyecto original de Dietmar Schröders. Lamentablemente, esta vez mi tiempo de desarrollo se vio limitado, empeorando más adelante debido a un pequeño infortunio con mi conversor USB/RS485 (véase el Labcenter de esta edición). Los lectores interesados deberían poder terminar el proyecto sin mayores inconvenientes, con algo de “know-how” sobre C, el código fuente del “medidor de gradiente de temperatura” y las herramientas del bus presentadas por ahora.

También tengo en mente ampliar el ElektorBusBrowser, de modo que pueda recibir mensajes sin ningún tipo de planificación. En principio, en esta aplicación sólo participan dos dispositivos. En cualquier caso,

Ajustar los intervalos de medida

El control remoto de un sólo medidor o sensor en principio no requiere planificación. Podemos establecer en el dispositivo cada cuánto queremos que transmita un nuevo valor de medida, como en las aplicaciones con dataloggers.



La gráfica muestra el correspondiente comando de 4 bytes del *ApplicationProtocol*. El valor del intervalo (Interval Value) se codifica mediante siete bits. En la escala de intervalos, cada uno significa:

Hex	Dec	Intervalo
04	4	µs
05	5	1/100.000 s
06	6	1/10.000 s
07	7	ms
08	8	1/100 s
09	9	1/10 s
0A	10	1 s
0B	11	10 s
0C	12	100 s
10	16	1 min
11	17	10 min
12	18	100 min
18	24	1 h
19	25	10 h
20	32	1 d
21	33	10 d
22	34	100 d
28	40	1 mes
30	48	1 año
31	49	10 años

Checksums y fiabilidad

Cuando se desarrolla un sistema de bus, uno nunca termina de preocuparse si los mensajes se transmiten correctamente bajo cualquier circunstancia. En la última entrega de la serie presentamos dos mecanismos de “acknowledge”. Uno, a nivel de los mensajes; pensado principalmente para evitar colisiones en FreeBusPhases no seguras. Y otro, a nivel del ApplicationProtocol, ideado para errores en las transmisiones y proceso de datos (incluyendo los nodos libres de colisión a los que se les pregunta, como ocurre en esta aplicación, véase el texto). En ambos casos, el receptor envía los bytes recibidos de vuelta al emisor, y se utiliza un sencillo bit de flag para diferenciarlo del mensaje original. Este método ha demostrado ser lo suficientemente seguro, y por eso se ha aplicado en lugar de un CRC o checksum.

Afortunadamente, ahora tenemos unos cuantos colaboradores del bus aportando ideas. El lector de Elektor Werner Koch cree que este mecanismo no es suficiente. Ciertamente, el emisor debería hacer reenvíos, en caso de que los mensajes se perdieran. Pero con ello no será inmune a perturbaciones que “generen” mensajes, que realmente no queremos que se envíen. A menos que un actuador envíe, por ejemplo, el correspondiente AcknowledgeMessage de vuelta al master; así podría detectarse si ha salido algo mal. Pero para entonces el actuador ya habrá activado el relé; podemos imaginar las consecuencias de esto.

Este problema puede solucionarse si el actuador tuviera que esperar por una confirmación desde el controlador, antes de activar el relé (“3 way handshake”). Otra posibilidad sería transferir los datos redundantes –sólo cuando el checksum se corresponde con los bits restantes, el actuador activa el relé. De aquí deducimos que así es mucho más improbable que se dé una coincidencia de este tipo, que en un mensaje sin protección alguna.

Ahora apareció la discusión de si íbamos a utilizar un CRC de 16 bits o un simple checksum. Mi propuesta de mantener los dos últimos bytes del mensaje con el valor AAhex fue rechazada de forma vehemente por algunos participantes de nuestra mail-list; aún así, era obvio que suponía una ventaja de cara a disponer de una sencilla sincronización. Otra de las cosas a favor de un CRC de 16 bits era eso mismo, la posibilidad de una sincronización más avanzada. Entre otras cosas, podíamos utilizar el CRC para determinar que un mensaje terminaba justo en ese punto.

Finalmente, hice la salomónica propuesta de combinar ambas posibilidades. Para diferenciarlas, utilizamos un bit de nuestro “modebyte”:

Bit	1	0
7	Sin bytes de ID, datos a partir del byte 2	Bytes de ID a partir del byte 2
6	Los bytes 2 y 3 son de ID	Los bytes del 2 al 5 son de ID
5	Sin CRC/Checksums	Bytes E y F son CRCs de 16 bits/Checksums
4	Sincronización avanzada	Sin AA_{hex} a partir del byte 2
3	El último byte de ID es un número dividido	Todos los bytes de ID para el direccionamiento
2	Los seis bits altos de direccionamiento para el segmento del bus	Sin direcciones del segmento
1	AcknowledgeMessage	Mensaje original
0	Se espera al AcknowledgeMessage	No se espera

Al menos una de estas comprobaciones será añadida a la librería en C para controladores AVR, que aún está empezando.

Un poco después todavía estaba pensando en implementar la redundancia dentro de los propios bytes de datos: ésta podría tratarse de uno, dos o cuatro bytes importantes que se transmiten repetidamente a lo largo del mensaje, por ejemplo de dos a cuatro veces. Para perturbaciones no periódicas, las probabilidades de que ese mensaje aparezca de la nada son prácticamente nulas.

el RS485 y los protocolos descritos resultan perfectamente aptos para el control remoto de un medidor. Ya que al sensor no le pregunta más un planificador, podemos programarlo de modo que sólo envíe los valores de medida en intervalos determinados. Los cuadros muestran cómo se puede codificar un comando de intervalos gracias al ApplicationProtocol.
En nuestra planificación tenemos aún más: en la próxima entrega, continuaremos con lo prometido, una aplicación para smartphones con Android, ¡y otras muchas que le seguirán!

(110610)

¡Forme parte del desarrollo! ¡Cualquier consejo o sugerencia son bienvenidos en el mail de nuestra redacción, redaccion@elektor.es!

Enlaces internet:

[1] www.elektor.es/110151
[2] www.elektor.es/110258
[3] http://ww1.microchip.com/downloads/en/DeviceDoc/21950e.pdf
[4] www.elektor.es/110428
[5] www.elektor.es/110610
[6] www.elektor.es/110517
[7] http://es.wikipedia.org/wiki/Hojas_de_estilo_en_cascada

Pendrive USB en un microcontrolador

Registrador para datos en serie

Thomas Fischl (Alemania) thomas@fischl.de

Este proyecto brinda la posibilidad de conectar directamente un pendrive USB a un sistema con microcontrolador. Mediante un stick USB de este tipo disponemos de una memoria no volátil fiable y asequible, capaz de obtener datos del microcontrolador, así como transportarlos y transferirlos. Sin embargo, para conectar el USB necesitaremos un controlador de host y para grabar, funciones de registro. Ambas tareas las solucionamos elegantemente gracias a un PIC24FJ64GB002 de Microchip.

Características:

- Compatible con USB 2.0
- Tasa de transferencia de hasta 12 Mbit/s
- Conexión con host: +5 V, GND, µC-Tx
- UART configurable mediante archivo en el propio pendrive
- Firmware open-source
- Alimentación: +5 V, de 50 a 80 mA

Incluso los pequeños microcontroladores disponen de un puerto serie (UART). Este puerto se utiliza a menudo para intercambiar parámetros de configuración o para analizar el estado actual de funcionamiento. Puede gestionarse la información con ayuda de un PC para el control. Sucede bastante que para testear un circuito han de tomarse datos durante largo tiempo, y así poder analizar el comportamiento de los componentes en funcionamiento continuo. Utilizar un PC para dicha tarea no siempre es posible, además, debido a su elevado consumo supone un gasto energético innecesario.

Para los microcontroladores actuales, la necesidad de integrar funciones lógicas y utilizar memorias suponen un gran inconveniente, principalmente porque las RAM o EEPROM sólo son aptas para el registro prolongado de datos bajo ciertas circunstancias.

La solución universal y con mayor rendimiento energético es el pendrive logger USB aquí presentado: registra todos los datos enviados por el sistema con microcontrolador por el puerto serie, guardándolos en un archivo del USB conectado.

Hardware

Básicamente, el circuito (**figura 1**) consta de un microcontrolador PIC24FJ64GB002 de Microchip. Este micro dispone de funcionalidad USB-2.0-OTG. OTG significa "On-The-Go" y se trata de una ampliación de las especificaciones del USB 2.0, que incluye una función limitada de host USB y permite intercambiar los roles entre dispositivo USB y dicho host. Este último también hace posible la comunicación entre dos dispositivos USB-OTG. En nuestro caso sólo necesitaremos la funcionalidad de USB host, que establece el enlace con el pen-

Productos y servicios Elektor

- Tarjeta 110409-1*
- Controlador PIC programado 110409-41
- Descarga gratuita del layout en PDF en [1]
- Descarga gratuita del firmware (archivo 110409-11 en [1])

drive USB mediante un conector USB tipo A normal, igual que los que podemos encontrar en un PC.
El núcleo del microcontrolador funciona a 3,3 V. Éstos se obtienen mediante el regulador de tensión IC2. El puerto serie se conecta con entradas que toleran hasta 5,5 V y protegidas por resistencias en serie de 220 Ω. Los 5 V de la alimentación del circuito también sirven para dar tensión al bus y alimentar a su vez al pendrive USB. Para protegerlo de cargas demasiado altas se ha colocado un fusible reemplazable en el propio conector USB.

El LED D1 y el pulsador S1 están conectados directamente al microcontrolador. El LED indica cuando hay una transferencia de datos. Con S1 se finaliza dicha transferencia. Para programar el microcontrolador, el circuito dispone de un puerto de programación K3 compatible con adaptadores conocidos de Microchip, como PICkit 3 o ICD2/ICD3. El jumper JP1 está pensado para posibles futuras actualizaciones del firmware. En la versión actual todavía no se utiliza.
Gracias a la tarjeta (**figura 2**) el montaje del circuito se simplifica en gran medida. Carece totalmente de componentes SMD, y todos se encuentran en la cara superior de la tarjeta. Para el microcontrolador, puede utilizarse un zócalo sin problema. La **figura 3** muestra el prototipo terminado de Elektor.

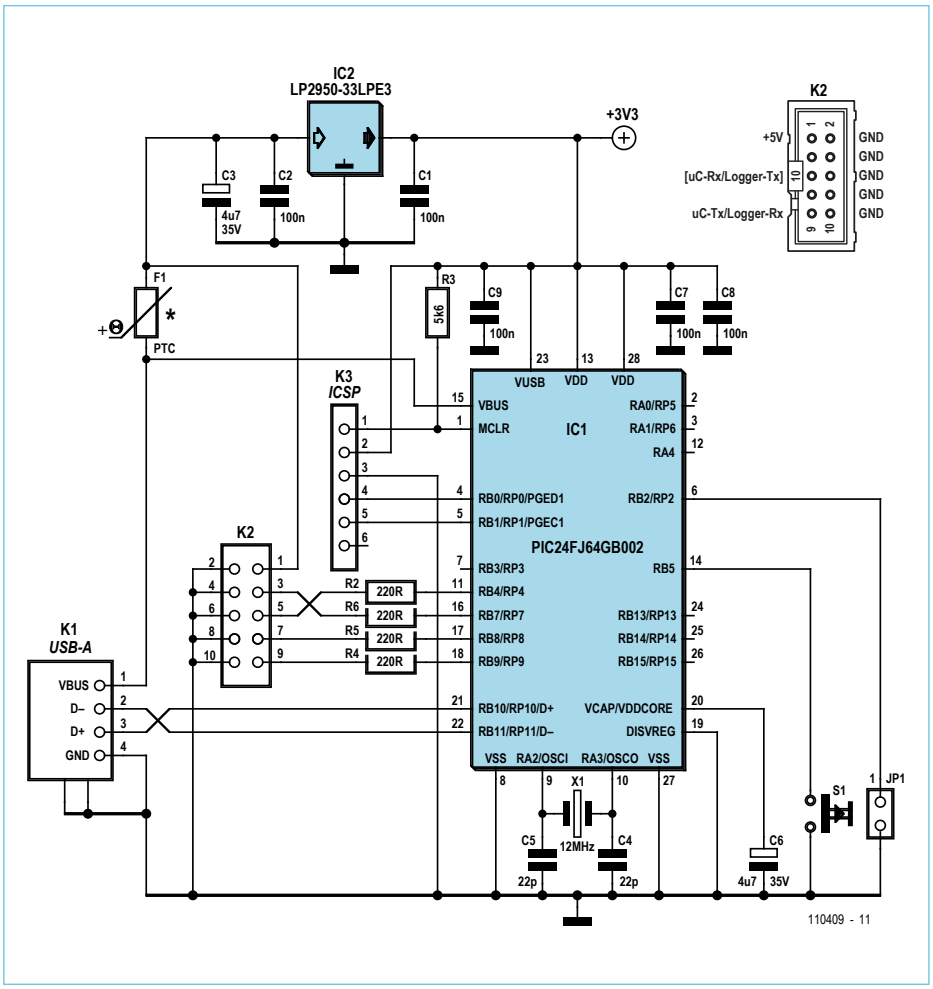


Figura 1. El circuito del pendrive logger USB para datos en serie consta principalmente de un microcontrolador con funcionalidad USB host.

Software

Microchip dispone de funciones para el control del USB gracias a las "Microchip Application Libraries". Entre otras, dan soporte a los dispositivos del tipo "Mass Storage Device", a los cuales pertenecen los pendrive USB. Lógicamente soporta el sistema de datos más común en estos sticks de memoria, el FAT. El firmware se codificó con MPLAB, el entorno de desarrollo de Microchip, y fue generado gracias al compilador C30. Todos los programas y librerías necesarios pueden descargarse gratuitamente, así como el firmware, en la página del proyecto de Elektor [1]. El firmware se transfiere al PIC mediante un programador, por ejemplo PICkit 3. Como alternativa, también puede utilizarse un controlador ya programado, ofertado en la página del proyecto [1].

El flujo de datos entre el UART y el sistema se ha implementado mediante dos buffers de ping-pong. Los caracteres recibidos por el puerto serie se almacenan en uno de los dos buffers. Una vez lleno completamente, la lógica del programa moverá todos los datos a la memoria del pendrive USB. Una vez transferida la información correctamente, el buffer se vacía y le "pasa el turno" a la unidad receptora.

Conexiones

La conexión entre el pendrive logger USB y el sistema microcontrolador que suministra los datos se hace mediante el puerto serie (UART). Los niveles lógicos pueden ser de un mínimo de 3 V y un máximo de 5,5 V. Si queremos utilizar el logger con un puerto RS232, tendremos que servirnos del conversor de nivel correspondiente (RS232/TTL). Para alimentarlo son necesarios +5 V, que en general ya existen en la mayoría de sistemas microcontroladores. El consumo depende del pendrive USB utilizado y está en un rango de 50 a 80 mA.
Todos los intercambios entre el logger USB y el microcontrolador se llevan a cabo mediante el conector K2. Actualmente las funciones de logging están limitadas únicamente a tres conexiones de K2: +5 V (pin 1), µC-Tx/Logger-Rx (pin 9) y masa (pin 10). Los pines 2, 4, 6 y 8 también van a masa.

Lista de materiales

Resistencias:
R1 = 1 kΩ
R2, R4 a R6 = 220 Ω
R3 = 5k6

Condensadores:
C1, C2, C7 a C9 = 100 nF
C3, C6 = 4µ7/35 V, vertical
C4, C5 = 22 pF

Semiconductores:
D1 = LED de baja corriente, 3 mm
IC1 = PIC24FJ64GB002-I/SP (programada: 110409-41)
IC2 = LP2950-33LPE3

Varios:
F1 = fusible reemplazable, 250 mA en régimen permanente, 500 mA en transitorio (Littlefuse 72R025XPR)
X1 = cuarzo de 12 MHz
K1 = conector USB del tipo A para montaje en tarjeta
K2 = conector de 2x5 pines, tipo pin header, acodado
K3 = conector de 6 pines, tipo pin header
Tarjeta 110409-1

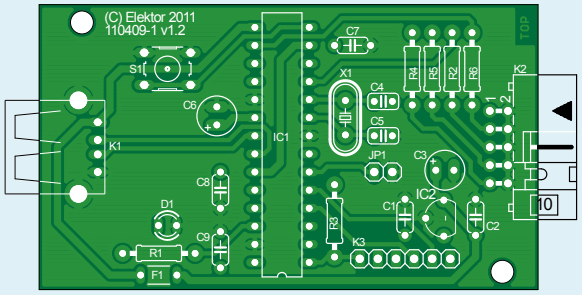


Figura 2. Montaje de la tarjeta, libre de componentes SMD.

La segunda línea de señal del puerto serie (µC-Rx/Logger-Tx) no se utiliza, sin embargo está disponible en el pin 5 para próximas actualizaciones del firmware, al igual que los pines de puerto 16 y 17 del microcontrolador, que se encuentran conectados a través de una resistencia de 220 Ω a los pines 3 y 7 de K2. Una posibilidad de ampliación sería, por ejemplo, la lectura desde el pendrive USB.

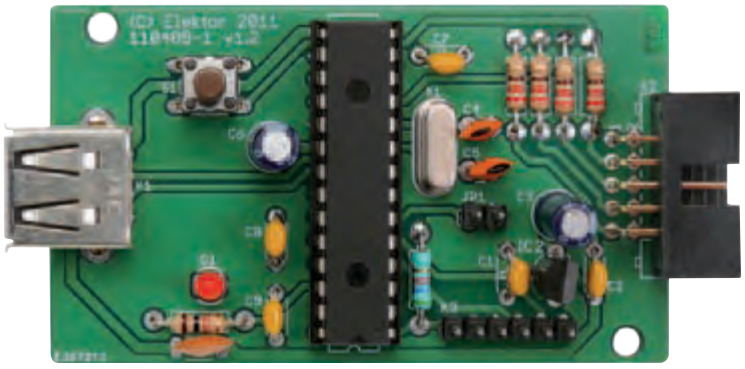


Figura 3. Tarjeta prototipo de Elektor montada.

Configuración

Los parámetros de la conexión serie simplemente se almacenan en un archivo de texto del pendrive USB: “config.txt”. Cuando se reconoce un pendrive (bien al activar la tensión de alimentación o al conectar un stick USB), se lee este archivo y el puerto serie se configura en consecuencia. Sin este archivo de configuración se fijan los siguientes valores estándar: 9600 baudios, 1 bit de start, 1 bit de stop, sin paridad.

Funcionamiento

Si el logger tiene tensión y hay un pendrive USB conectado, comenzará el modo de escritura. Un breve parpadeo del LED señaliza que se están recibiendo caracteres por el puerto serie. Los datos recibidos se almacenan 1:1 en el archivo “logging.txt”. Antes de desconectar el pendrive USB hemos de presionar el pulsador. Después, todos los caracteres escritos en el buffer y el archivo de logging se cerrarán correctamente. Los datos registrados pueden leerse con cualquier PC; sencillamente conectamos el pendrive y abrimos “logging.txt” con un editor de texto corriente.

Sugerencias

Junto con la función de logging desarrollada en el presente circuito, podrían desarrollarse expansiones del firmware, pero no sólo en cuanto al stick USB, sino también por ejemplo para seleccionar los datos. Otra posibilidad sería desarrollar un logger “stand-alone”, que cuente con entradas analógicas y digitales, y seleccione automáticamente los intervalos de guardado en el pendrive USB. En K2 hay en total cuatro líneas de datos del microcontrolador PIC, y con una matriz interna pueden conectarse con distintas señales de periféricos, y diseñar por ejemplo un puerto SPI o un UART adicional.

(110409)

Enlace:

[1] www.elektor.es/110409

E-Blocks en Twitter

Usando redes inalámbricas embebidas

Ben Rowland (Rusia)

En este proyecto vamos a ver cómo podemos conectar fácilmente una tarjeta de red inalámbrica a nuestro sistema microcontrolador para desarrollar una página web que contenga información útil sobre el entorno e incluso envío de mensajes en Twitter.

En nuestro club náutico local hay más de 1000 miembros. Una de sus dificultades es que no saben cuando las condiciones meteorológicas son adecuadas para la navegación. No se trata sólo de saber qué viento tiene que haber para navegar, sino que las regulaciones de salud y seguridad dictan que un socorrista cualificado (uno de los miembros) debe estar presente cuando alguien está en el lago. Para resolver este problema, propusimos que se podría crear una página web para informar a los miembros cada vez que el socorrista entra o sale de su puesto en el club náutico, así como las condiciones climáticas locales y otras informaciones de navegación. También debía haber en la página web un enlace a la popular red social Twitter [1], por lo que un miembro puede hacer saber a otros que está yendo hacia el club y que las condiciones son adecuadas. De igual forma se propuso que la página web debía incluir una cámara web, montada sobre un servomotor controlado por el visitante de la web. Idealmente, para proporcionar información visual, el servomotor debe responder también a los comandos de control del visitante de la página web.

Los elementos usados

Para que el proyecto esté en marcha y funcionando en el banco de trabajo, he usado una selección de "E-blocks" que se pueden ver en la **Figura 1**. El conjunto consiste en un Multiprogramador EB006, equipado con un PIC18F4455, una placa de Sensores EB003, una placa de Conmutación EB007, una placa LCD EB005, una placa Interfaz Servo EB059 y una placa de red LAN inalámbrica EB069. En beneficio del prototipo, la lectura de la temperatura proviene de una sonda de temperatura de acero inoxidable, insertada en la placa de sensores. Las lecturas de la luz y la velocidad del viento llegan desde la LDR y el potenciómetro, también de la placa de sensores. Por último, el sensor del socorrista es simplemente un interruptor (SW0) en la placa EB007. Este conmutador podría incluso, eventualmente, estar colocado bajo el asiento del socorrista, para permitir que los mensajes de Twitter se envíen de forma automática sin intervención del usuario.

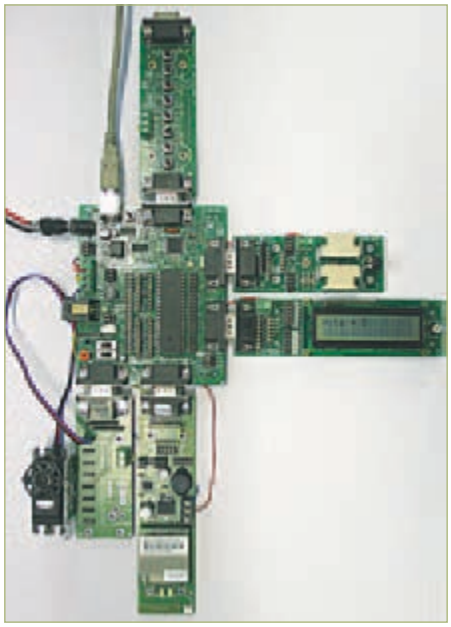


Figura 1. El sistema prototipo basado en E-blocks.

La placa de E-Blocks, Matrix Multimedia Wireless LAN, es un elemento clave del sistema: dicha placa permite un acceso sencillo a la red inalámbrica del club náutico y está totalmente soportaba por Flowcode V4. El E-block puede ser usado para alojar una red inalámbrica o unirse a una red inalámbrica existente. En el modo de red huésped ("host") no hay un

Productos y Servicios de Elektor:

- Multiprogramador E-Block (EB006)
  (Nota: PIC18F4455 no incluido)
- Placa Sensor E-Block (EB003)
- Placa Conmutador E-Block (EB007)
- PlacaLCD E-Block (EB005)
- Placa Interfaz Servo E-Block (EB059)
- Placa LAN Wireless E-Block (EB069)
- Flowcode para dsPIC/PIC24: #TEDSSI4
- Fichero de programa Flowcode: 110388-11.zip (ver [2])

Precio y detalles de pedido en www.elektor.com/e-blocks

modo sencillo de permitir el acceso a Internet, por lo que el propósito de este artículo será el de usarlo en modo cliente.

Configurando la placa LAN sin hilos

En este artículo queremos ser capaces de comunicarnos con el sistema de E-Blocks a través de Internet, por lo que, para comenzar, lo primero que necesitamos es conectar el bloque electrónico inalámbrico a la red inalámbrica local existente, como se muestra en la **Figura 2**.

La placa WLAN puede actuar como un servidor, proporcionando páginas inalámbricas a otros dispositivos LAN inalámbricos o como un dispositivo cliente que se comunica con un servidor remoto. Para comenzar este proceso, primero tenemos que configurar el elemento LAN inalámbrico para que se comporte como un dispositivo final. Para la mayoría de los sistemas, la configuración de las propiedades del componente WLAN de Flowcode mostrado en la **Figura 3** será el correcto.

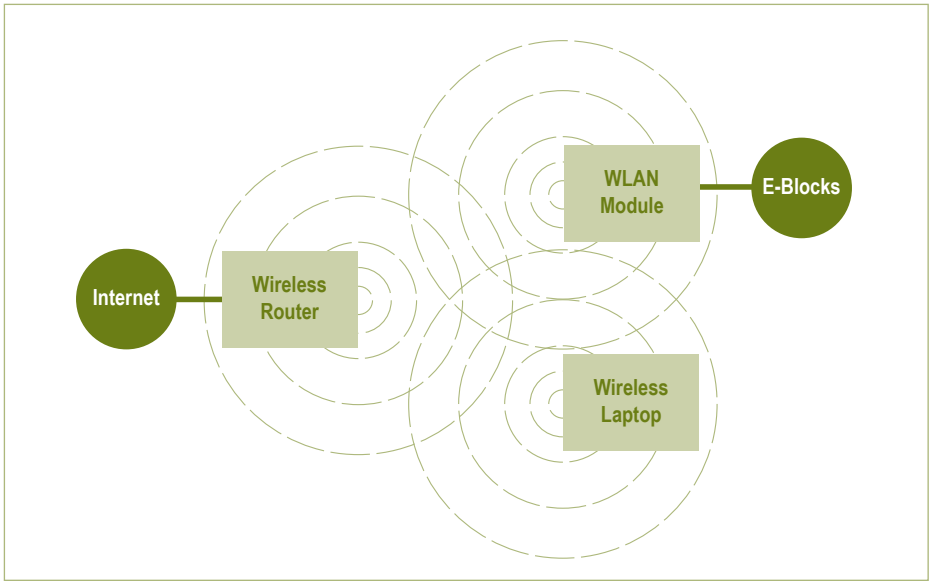


Figura 2. Configuración de la Red de Área Local (LAN) inalámbrica o “wireless”.

Para permitir peticiones de Internet en la WLAN tenemos que inicializar el módulo WLAN, lo que implica reiniciar el dispositivo y transferirle el conjunto de propiedades de inicialización de Flowcode. En la **Figura 4** podemos ver la rutina que hace este proceso. Una vez que hemos hecho esto, se usa la macro con el componente Connect_To_SSID para conectar el módulo al router inalámbrico huésped (“host”). La macro de conexión requiere dos parámetros: el primero es el nombre de la red (SSID) y el segundo es la clave de seguridad inalámbrica. Si la red no tiene seguridad, cualquier conjunto de caracteres puede ser utilizado como clave. Podemos ver este programa en la **Figura 5**.

Configurando nuestro router

Una vez que el sistema está funcionando y en ejecución, deberíamos poder ver las páginas web servidas por el sistema embebido en la red local. Para ver las páginas sobre la red local primero tenemos que descubrir la dirección IP del módulo WLAN, la cual debe ser mostrada en la lista de clientes DHCP de nuestro router. La introducción de la dirección IP de nuestro módulo WLAN en un navegador de Internet nos mostrará la herramienta de configuración WLAN. Es similar a la herramienta de configuración de un router estándar y nos permitirá verificar todas las configuraciones de Flowcode que han sido volcadas en el módulo de forma correcta. Para ver las páginas de datos reales servidas por el sistema necesitamos añadir manualmente el puerto específico del servidor a nuestro navegador URL. Aquí tenemos un ejemplo de dirección URL donde la dirección IP del módulo WLAN es 192.168.0.4 y el puerto del servidor es 5000:

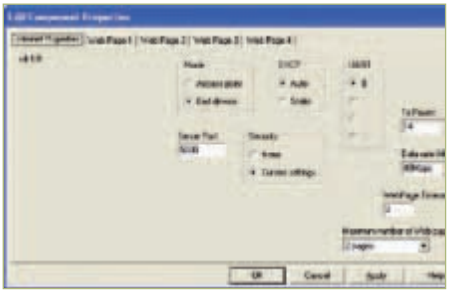


Figura 3. Diálogo de Flowcode para LAN inalámbrica.

http://192.168.0.4:5000/

Conexión a Internet

Una vez que hemos confirmado que el módulo WLAN está “sirviendo” páginas correctamente, podemos configurar el router para permitir que el módulo pueda ser direccionado a través de Internet. El hacer esto significa que podemos acceder al sistema embebido desde cualquier lugar del mundo. Para ayudar a configurar nuestro router específico, hay una página web en http://portforward.com, que nos guía a través de una serie de pasos que necesitamos hacer para permitir el acceso web al sistema embebido. También disponen de un servicio

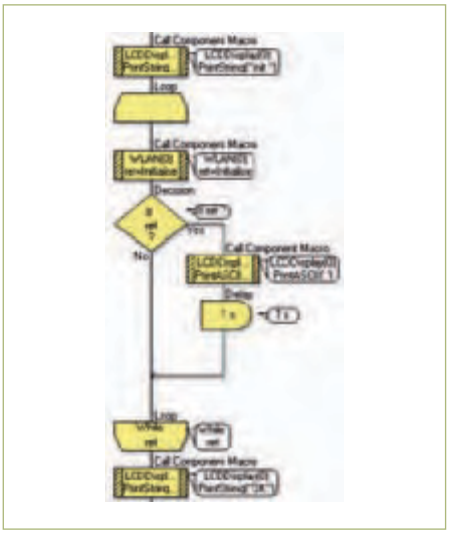


Figura 4. Inicialización de WLAN en Flowcode.

de pago para ayudarnos a arrancar y poner en funcionamiento nuestro sistema. Nuestro manual del router también contiene una buena fuente de información de cómo hay que hacer esto para nuestros dispositivos.
Para conectar con el sistema embebido a través de Internet tenemos que introducir la dirección URL de nuestra conexión local de Internet como lo detalla nuestro router. Como la dirección IP proporcionada por nuestro proveedor de servicios de Internet puede cambiar regularmente, hay servicios gratuitos, como http://no-ip.com, que nos proporcionarán un nombre de dominio estático gratuito que, automáticamente nos devolverá a nuestra dirección IP real. El módulo WLAN soporta directamente este tipo de funcionalidad, denominada Dynamic Domain Name System (DDNS, o Sistema de Nombres de Dominio Dinámico), de manera que podemos introducir nuestro nombre de usuario y contraseña no-IP en la herramienta de configuración del módulo y, ésta guardará automáticamente nuestra dirección IP "sincronizada" con nuestro nombre de dominio.

Configuración de las páginas web en Flowcode

El contenido de la página web se configura introduciendo código HTML y Javascript directamente en el componente WLAN de Flowcode. Podemos ver un ejemplo de ello en la **Figura 6**. Las variables usadas en la página web, como temperatura y velocidad del viento, enlazan directamente con las variables del programa Flowcode. La salida de información de las variables se controla utilizando un componente macro de Flowcode e insertándolo en el código HTML usando el carácter de '%' seguido por un número de índice.
Ejemplo, `Temperature = %0`.

Por otro lado, la entrada de las variables está controlada por el índice y valor de variables añadido a la dirección URL, similar a como se pasan las variables en PHP.
Ejemplo, `index.htm?0=255&1=39`

Las solicitudes de página son atendidas regularmente llamando al componente macro Check_For_Page_Requests, dentro del programa Flowcode. Ahora ya disponemos de un sistema microcontrolador versátil que se puede comunicar de forma inalámbrica con redes locales e Internet por igual, y ya somos capaces de pasar los valores de entrada y salida del sistema. En las **Figuras 7** y **8** podemos ver ejemplos de páginas web atendidas por el microcontrolador. La página principal muestra la información del tiempo y un enlace a una sub-página que permite a los usuarios controlar la dirección de la cámara. La página principal también muestra un enlace a la red social Twitter que puede ser usado por un miembro para enviar un mensaje 'presencial' al resto de miembros suscritos en Twitter.

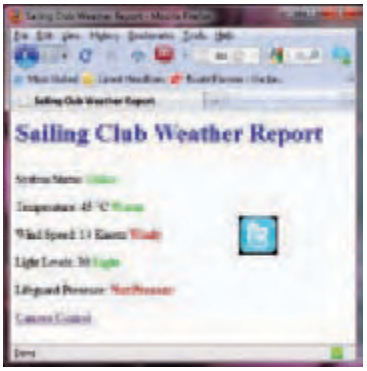


Figura 7. Informe del tiempo en Club náutico.

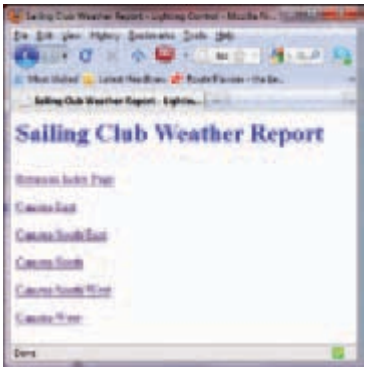


Figura 8. Control de cámara del Club náutico.

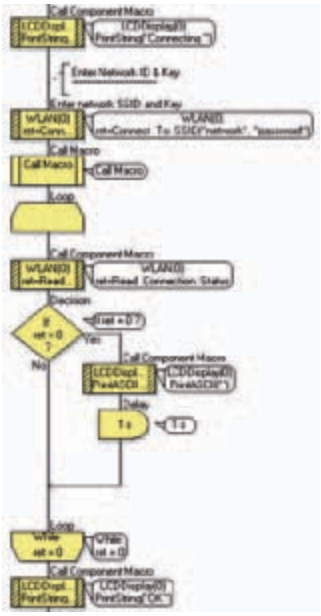


Figura 5. Conexión de WLAN a SSID.

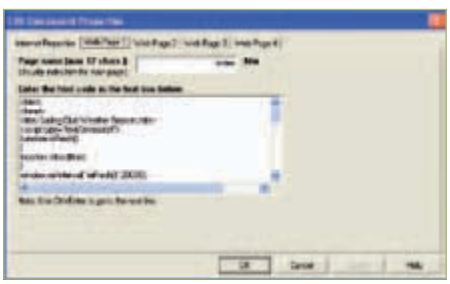


Figura 6. Página HTML de Flowcode para LAN inalámbrica.

Creando un enlace en Twitter

Lo que restabo era el mensaje de Twitter detallando lo que está sucediendo en el sistema para todos los miembros conectados. Esto se hizo creando un botón de Twitter en la página web y rellenándolo con los datos recogidos por los sensores. Después, cuando los usuarios visitan la página web y pulsan en el botón de Twitter, pueden enviar un mensaje a cualquiera de sus seguidores detallando las condiciones climáticas en el club. Traté de conseguir que el sistema enviase automáticamente los mensajes de Twitter cada vez que el socorrista entraba o salía del club, pero no pude conseguir que funcionase de forma fiable, por lo que lo he abandonado, por el momento.

Conclusión

Todo este kit está ahora en funcionamiento en el banco y se comunica claramente con la web. El siguiente paso es hacerse con un anemómetro y llevar el montaje para su uso en campo ... Los programas de Flowcode están (como siempre), disponibles aquí, en el sitio web de Elektor [2].

(110388)

Enlaces en Internet y Literatura

[1] www.twitter.com
[2] www.elektor.es/110388
http://portforward.com
http://no-ip.com

Código de bloqueo a toda prueba

Usando encriptación AES de 128 bits

Elbert Jan van Veldhuizen (Holanda)

¿Cómo es de seguro el control remoto de nuestro coche u otro vehículo valioso? Este proyecto nos muestra cómo utilizar un par de microcontroladores ordinarios, junto con un transmisor y un receptor, para implementar un sistema de mando a distancia por infrarrojos (IR) que usa un código de seguridad con un cifrado AES de 128 bits y una comunicación de datos IR bidireccional.

Todos estamos familiarizados con los sistemas de control remoto de bloqueo de las puertas en los coches. El mando a distancia transmite un código y, si el receptor del sistema de bloqueo reconoce el código correcto, desbloquea las puertas. Un inconveniente de los sistemas de este tipo es que personas con intenciones maliciosas que espían la transmisión del código, pueden capturar el código y, posteriormente, transmitir el código en cualquier momento que deseen para desbloquear el coche.

Un método mucho más seguro es el protocolo de autenticación de aceptación "desafío-respuesta", el cual se utiliza de muchas maneras en la banca por Internet. Con este método, la unidad de bloqueo transmite un código específico y el control remoto debe realizar un cálculo definido usando este código. El resultado se envía de vuelta a la cerradura. La unidad de bloqueo se mantiene bloqueada a menos que se haya realizado el cálculo correcto.

Escuchar clandestinamente durante la comunicación entre la unidad de bloqueo y el control remoto no sirve para nada en este caso, ya que la siguiente vez la unidad de bloqueo enviará un código diferente para el cálculo. Siempre y cuando la persona con malas intenciones no conozca el cálculo, la cerradura no será descifrada.

Por eso es importante elegir un buen método de cálculo. El cifrado es muy adecuado para ello. El cifrado utiliza una clave para convertir los datos en nuevos datos, que es exactamente el tipo de cálculo que necesitamos para el sistema de control remoto.

Algoritmo de cifrado

¿Qué hace que un algoritmo de cifrado sea bueno? El cifrado es un proceso en el cual los datos que tienen que ser cifrados (*in*) se convierten en datos cifrados (*out*) con la ayuda de una *clave*, que es algo así como una contraseña.

Esto también se aplica a la clave: dos claves diferentes producen dos valores únicos de cifrado.

- El algoritmo de cifrado implementa la función *out* = f(*in, key*). La función inversa *in* = f_{inv}(*out, key*) también existes, pero la función *key* = f_{key}(*in, out*) no existe.
- Para cada valor de *in* existe un valor único *out*. En otras palabras, no hay múltiples valores de *in* que generen el mismo valor de *out*.
- Esto también se aplica a la clave (*key*): dos claves diferentes producen dos valores únicos encriptados de salida (*out*).

La primera condición garantiza que si las personas con intenciones maliciosas llegan a conocer los valores de *in* y *out* espiando la señal, no serán capaces de calcular la clave usando la función f_{key}. La única manera de determinar la clave es la de utilizar lo que se llama un "ataque de fuerza bruta", que consiste en intentar todas las claves posibles de la función f de encriptación. Esto requiere gastar una gran cantidad de tiempo buscando la clave con lo que el intentar probar todos los posibles valores llevaría demasiado tiempo. Los ordenadores potentes pueden probar todos los posibles valores de una clave de 64 bits en, aproximadamente, un día. Con una clave de 128 bits, esto tar-

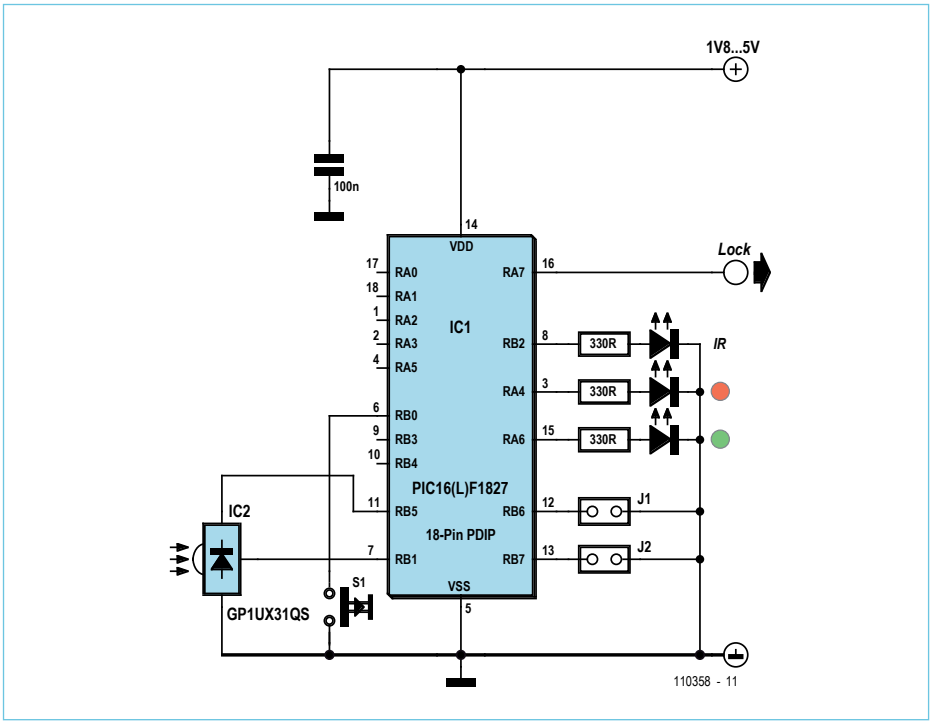


Figura 1. Esquema eléctrico de la estación base.

daría más de la vida del universo, tiempo suficiente para estar seguros.

Hemos elegido el protocolo AES para el algoritmo de cifrado. Este algoritmo de cifrado se utiliza en dispositivos como routers Wi-Fi para impedir la captura de tráfico de datos y la entrada en la red. Este algoritmo no ha sido descifrado ("craqueado") hasta ahora. En otras palabras, nadie ha encontrado la función f_{key} o alguna otra manera de determinar la clave correcta con pocos intentos. El algoritmo AES necesita gran cantidad de recursos en comparación a lo que es habitual en el ámbito de los microcontroladores. Son necesarios unos 240 bytes de memoria RAM para los cálculos, el código contiene, aproximadamente, 1.500 instrucciones, y la ejecución del cálculo lleva, aproximadamente, 30.000 ciclos de instrucción. Además, se utilizan ampliamente las tablas y los arrays.

El nuevo microcontrolador PIC16F1827 mejorado de Microchip es un dispositivo que cumple con estos requisitos. Tiene 4096 palabras de memoria de código y 396 bytes de RAM. También puede funcionar a velocidades de hasta 32 MHz usando su reloj interno, y el microcontrolador cuenta con un nuevo conjunto de instrucciones llamado "rango medio mejorado", que le permite trabajar con arrays de manera mucho más fácil. Aunque la memoria RAM se divide en bloques individuales de 80 bytes cada uno, el conjunto de instrucciones mejorado permite que los bloques individuales puedan ser visualizados como un bloque único de gran tamaño (modo lineal), lo que facilita el acceso a las tablas.

El circuito

Aparte de los microcontroladores, sólo son necesarios unos pocos componentes para implementar los circuitos (transmisor y receptor) que utilizan este método de cifrado. La **Figura 1** muestra el esquema eléctrico del circuito de la estación base (la llave), mientras que la **Figura 2** muestra el esquema eléctrico del circuito del control remoto. Ambos circuitos se comunican mediante diodos LED de infrarrojos, de la misma forma que los mandos a distancia de los televisores. El diodo LED D1 se utiliza para la transmisión, mientras que IC2 (un módulo estándar de IR de la casa Sharp con una frecuencia de operación de 36 kHz) se utiliza para la recepción. Los interruptores para el funcionamiento de los dispositivos y la configuración de los parámetros están conectados directamente a los puertos de E/S. La característica de "pull-up débil" ('weak pull-up') del microcontrolador hace que las resistencias sean innecesarias aquí. También se puede conectar un teclado al mando a distancia para introducir un código PIN. Para este fin es necesario utilizar un teclado de matriz.

El control remoto funciona directamente con dos pilas AAA, pero también se puede utilizar una pila botón de litio. La estación base puede ser alimentada por un adaptador de tensión de red. El rango de tensión de alimentación permitido es de 1,8 a 5 V. Hay que tener en cuenta que la tensión nominal máxima de la versión LF del microcontrolador es de 3,3 V.

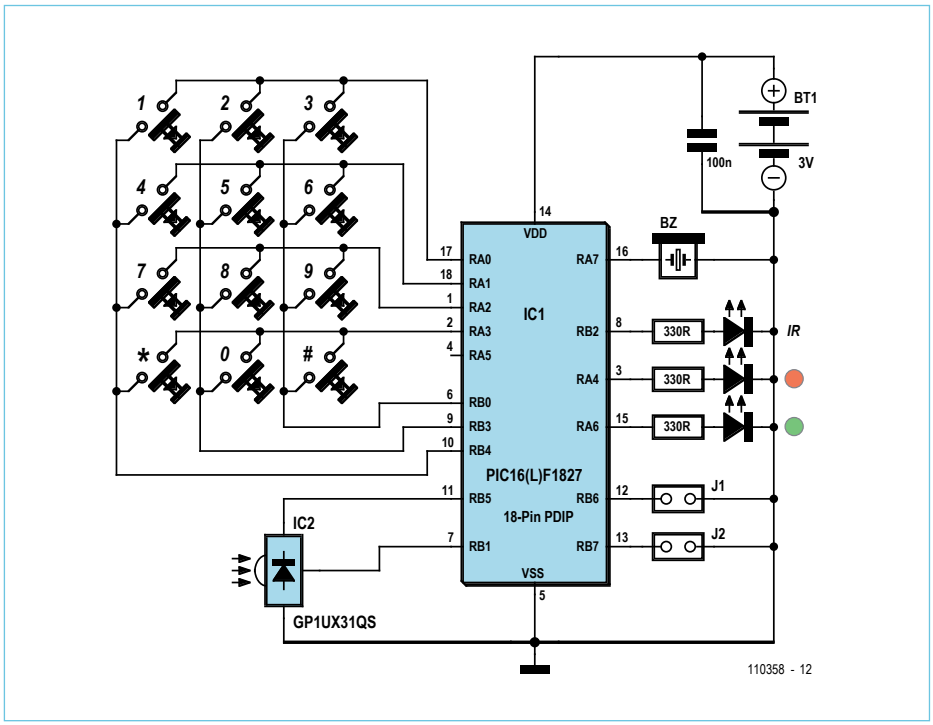


Figura 2. Esquema eléctrico del control remoto.

Funcionamiento

Una sesión de comunicación se inicia cuando el mando a distancia envía el código 'A6h". La estación base genera un número aleatorio de 128 bits. Un número aleatorio es mejor que un número predecible ya que la clave puede ser capturada, potencialmente, por un "cazador de código" si se utiliza un número predecible. El algoritmo de encriptación es también un excelente generador de números aleatorios (ver recuadro), usando un valor de entrada procedente de un contador. El algoritmo de cifrado convierte el valor de entrada en un número aleatorio (usando una clave independiente). El valor del contador se guarda en la memoria flash para que se puedan generar números únicos después de un apagado del equipo. Como la memoria flash tiene una vida nominal máxima de 100.000 operaciones de escritura, el valor se guarda en la memoria sólo una vez cada 65.536 veces, y en una ubicación de memoria diferente cada vez que se utiliza. En el improbable caso de que se alcanzara el número máximo de operaciones de escritura (des-

El uso de encriptación para la generación de números aleatorios

Los registros de desplazamiento con realimentación lineal se utilizan comúnmente para generar números aleatorios. Su trama de bits de salida tiene las características estadísticas de aleatoriedad, pero las tramas de bits son predecibles. A medida que se va conociendo el algoritmo, el estado del registro de desplazamiento se puede reproducir después de haber sido leído un conjunto de datos específicos. Esto permite que los valores puedan predecirse.

Un buen algoritmo de cifrado también tiene las características estadísticas de aleatoriedad. Debido a la asignación única de la entrada hacia la salida, la relación de unos y ceros será exactamente del 50%. Sin embargo, la trama de bits es totalmente impredecible ya que la clave no se conoce. El patrón se repite por sí mismo (o la clave puede ser determinada mediante el cálculo) sólo después de que la trama de bits completa haya sido generada (en este caso 2^{131} bits). Si esta trama de bits se transmite a una velocidad de 1 Gbit/s, se necesitará un trillón de veces la vida del universo para transmitir la trama de bits completa.

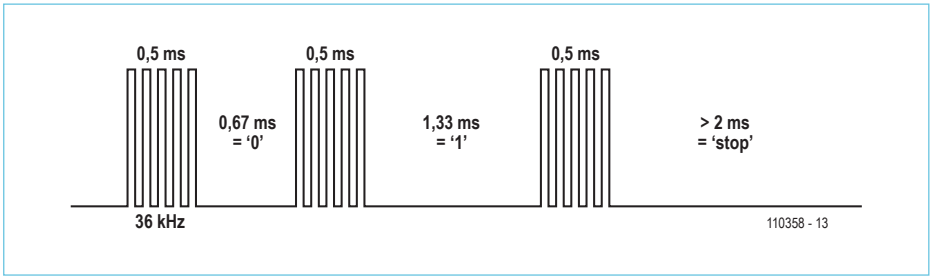


Figura 3. Se ha utilizado una variante del protocolo Sharp para transmitir los datos usando pulsos de IR.

pués de 13 millones de interrupciones de la alimentación o de 900 millones de transacciones), se activa un procedimiento de emergencia para garantizar que el usuario no se quede de pie, delante de una puerta bloqueada. Este procedimiento requiere que el usuario presione el botón del mando a distancia 16 veces seguidas. Después de esto, el número aleatorio se obtiene de la temporización de la transmisión del código por el mando a distancia.

Comunicación por infrarrojos

El control remoto primero lee el número de 128 bits. Los módulos estándar pueden manejar solamente un ciclo de trabajo máximo del 30% para transmisiones de este tipo. El método de codificación Manchester utilizado comúnmente (usado en el protocolo RC5, por ejemplo), tiene un ciclo de trabajo del 50%. Por esta razón, aquí se utiliza una variante del protocolo de Sharp. Los valores '1 'y '0' se definen por la longitud de la separación entre dos pulsos. Una separación de 0,67 ms es un '0 ', mientras que una separación de 1,33 ms es un '1'. El ancho de pulso es de 0,5 ms, y el final del tren de pulsos viene indicado por una separación que dura más de 2 ms (ver **Figura 3**). Las tolerancias de temporización se pierden y el algoritmo se auto-sincroniza, por lo que la precisión del reloj oscilador no tiene por qué ser especialmente alta. Este protocolo también se puede utilizar para transmitir palabras de 8 bits (o palabras de cualquier longitud), con la misma facilidad que las de 128 bits, gracias a la utilización de un bit de "stop" (fin o parada).

Tanto el mando a distancia como la estación base aplican el cifrado al número aleatorio de 128 bits, utilizando la misma clave. El mando a distancia envía el número de 128-bit encriptado a la estación base. La estación base compara el número recibido con el calculado por ella misma. Si coinciden, la estación base abre la cerradura. Dependiendo de la configuración de un puente 1, la estación base puede devolver un código que indica coincidencia del cálculo (0xab) o ninguna coincidencia (0xB5) en el mismo. En teoría, la devolución del código resultante permitiría la captura de la clave usando un método automatizado aunque, en la práctica, esto es bastante improbable. Sin embargo, si consideramos el riesgo demasiado grande, podemos montar el puente 1 para evitar la transmisión de los códigos resultantes.

El puente 1 de la placa del mando a distancia tiene una función similar: el mando a distancia emite un pitido de tono bajo, si el código es incorrecto, o la estación base no envía una respuesta. La colocación de puente desactiva este tono de aviso.

La generación de la clave

El puente 2 permite la programación de la clave. Esto requiere apagar la estación base y volver a encenderla. Si pulsamos ahora S1 32 veces seguidas, se generarán dos claves. El diodo LED rojo se apagará levemente cuando este proceso se haya terminado. La temporización de la pulsación de botón proporciona unos números totalmente aleatorios debido a la velocidad del contador. Estos números son almacenados en la memoria EEPROM. Después de esto, se debe programar el control remoto con la misma clave. Para ello, el puente 2 de la placa del mando a distancia debe estar también montado. Después de esto, el mando a distancia envía el código '0xAD' (es posible que tengamos que pulsar la tecla # o introducir el código PIN primero). A continuación, la estación base envía la clave dos veces. El mando a distancia verifica que los dos números transmitidos son los mismos y, posteriormente, guarda la clave en la EEPROM (el diodo LED verde se enciende y suena un pitido). Esto se puede repetir con cada unidad de control remoto. Retire los puentes y apague las unidades para restaurar la estación base y el control remoto a su modo de funcionamiento normal.

Por razones de seguridad, la clave sólo se puede enviar al control remoto inmediatamente después de que haya sido generada en la estación base. Esto evita la programación 'clandestina' de otro mando a distancia más tarde. Esto sólo se podría hacer generando una nueva clave, con el resultado de que el control remoto original ya no funcionaría, con lo que esta acción siempre será detectada.

Además, la protección de datos de la EEPROM y la memoria del programa está habilitada tanto en la estación de base como en el mando a distancia. Esto significa que la clave nunca se puede leer. Además, la clave no se conoce cuando se genera, ya que el usuario simplemente pulsa el botón, sin conocer el valor de la clave que se genera de esta manera. La clave es almacenada de forma segura en el microcontrolador.

Sin embargo, existe un riesgo: cuando tanto

la estación base como el mando distancia son nuevos a estrenar (aún sin programar), el sistema ya está operativo debido a que ambos dispositivos tiene sus memorias EEPROM rellenas con '0xFF', con lo que ambos microcontroladores tienen la misma clave. Un usuario puede pensar que no hay necesidad de programar la clave y alguien con malas intenciones podría utilizar la clave 'FF...FF' para intentar abrir cerraduras de este tipo. Para evitar esto, la unidad de control remoto (pero no la estación base) siempre incrementa la clave leída desde la EEPROM, por lo que las claves no son las mismas. Cuando la clave es programada el valor se decrementa con el último resultado de clave correcta que se usó.

Código PIN

El control remoto está equipado con un teclado para introducir un código PIN. El código PIN está desactivado por defecto. Si no deseamos usar un código PIN, basta con conectar un pulsador entre RA3 y RB0 (lo que corresponde a la tecla # en el teclado). El código PIN puede ser configurado montando el puente 2 y pulsando la tecla * (o introduciendo el código PIN actual si ya ha sido configurado). A continuación, introduciremos el código PIN nuevo dos veces seguidas. Para desactivar el código PIN, se debe establecer un nuevo código PIN con un valor de #### o ****.

Si se ha configurado un código PIN, éste debe ser introducido en el mando a distancia cuando está activado. Si se introduce un código PIN de forma incorrecta tres veces, el mando a distancia se bloquea borrando la clave. Después de esto, el mando a distancia debe ser re-sincronizado con la estación base mediante la generación y programación de una nueva clave.

On/off

Hemos podido darnos cuenta de que el control remoto no tiene un interruptor de encendido. El "problema" aquí es que el consumo de energía de los últimos microcontroladores PIC es tan bajo que el circuito no se apaga de inmediato, debido a la energía almacenada en el condensador de desacoplo. Por esta razón, se optó por una solución diferente. Transcurridos cinco segundos, el mando a distancia entra en el "modo dormido" y, en este modo, su consumo de corriente es prácticamente cero (mucho menos que la tasa de auto-descarga de las pilas). La tecla # genera una interrupción y es esta interrupción la que genera en "encendido" del mando a distancia.

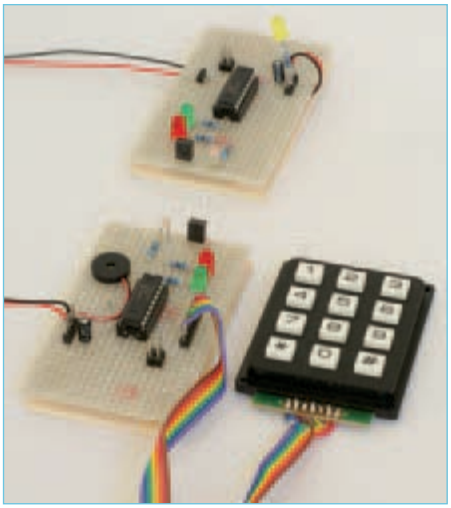


Figura 4. Ambos circuitos pueden construirse fácilmente usando piezas de placas de prototipo.

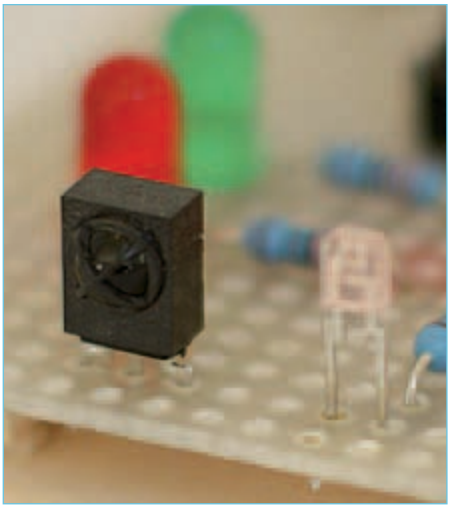


Figura 5. El LED de IR y el receptor de IR están colocados cerca el uno del otro, enfrentados en la misma dirección.

El programa

El autor convirtió el código fuente abierto existente C++ para las rutinas AES (el fuente está incluido en el código), en lenguaje ensamblador ya que el código no se compilaba correctamente con los compiladores C para microcontroladores PIC. El código del programa para la estación base y el control remoto se encuentra en un archivo único, ya que muchas rutinas son las mismas para ambos dispositivos. El fichero hexadecimal correcto se puede generar mediante la colocación de "# define remote" o "# define homestation" al principio del código. Naturalmente, el código también se puede modificar.

El diodo LED de infrarrojos y el sensor de infrarrojos están conectados al puerto serie (TX/RX). Otros dispositivos, tales como un módem GSM, también se podrían conectar a este puerto. Esto permitiría que una llave (u otro dispositivo) pueda ser activado de forma segura en cualquier parte del mundo mediante el envío de mensajes de texto.

El interfaz de usuario

El uso de los dispositivos es sencillo, una vez que la clave ha sido programada como se ha descrito anteriormente.

- Pulse la tecla # en el mando a distancia para activarlo. Si el diodo LED parpadea rápidamente, se debe introducir el código PIN. Si un código PIN se introduce tres veces, el diodo LED rojo parpadeará constantemente y el mando a distancia debe ser re-sincronizado.
- En este punto podemos tener tres resultados diferentes de pitidos/visuales:
- Un tono de pitido alto/LED verde – tono de pitido bajo/LED rojo: la estación base no responde (puede estar demasiado lejos o no estar encendida).
- Un tono de pitido alto/LED verde – tono de pitido bajo/LED verde: clave errónea; puerta permanece bloqueada (o el mando a distancia puede estar bloqueado).
- Un tono de pitido alto/LED verde – tono de pitido alto/LED verde – tono de pitido alto/LED verde: puerta abierta.
- Si el puente 2 está montado, sólo se produce la secuencia "tono de pitido alto/LED verde – tono de pitido alto/LED verde" en las dos últimas situaciones. En este caso, la apertura de la cerradura (o no) es la única indicación de si se ha usado la clave correcta.
- La tecla # se puede pulsar de nuevo, dentro de 5 segundos, para enviar una nueva petición de desbloqueo sin necesidad de introducir de nuevo el código PIN.
- El control remoto se apaga automáticamente después de 5 segundos sin actividad del usuario.

(110358)

Atención: Si conseguís descubrir una manera de revelar el código de esta llave codificada, por favor, hacédselo saber a los editores: editor@elektor.com

Time-lapse con una tableta Android

Un servo controla el botón de disparo

Elbert Jan van Veldhuizen (Holanda)

Con la ayuda de una tableta Android y un poco de hardware puedes controlar una cámara de fotos, con la que puedes hacer una serie de fotografías en un lapso de tiempo. También permite realizar ajustes en las tomas a través de un navegador Web del PC, para enviarlas luego a la tableta a través de la red local WiFi.

Este diseño controla una cámara que saca fotos con regularidad. Combinando luego estas fotografías en un video se obtiene una secuencia de vídeo acelerada (time-lapse), donde se muestran las horas o días en segundos. En una edición anterior de Elektor se publicó un proyecto parecido para cámaras de fotos con una entrada externa para el botón de disparo. Este proyecto controla el botón de disparo mediante un servo (RC). Lo especial de este proyecto es la utilización de una tableta (barata) Android para la GUI y el control a través de Internet, tal y como se describe en otra parte de esta edición. Este artículo también es interesante para comprender mejor los elementos básicos necesarios para la programación de tabletas Android en sistemas embebidos.

Diseño y control

En este diseño se optó por controlar mecánicamente una cámara de fotos con ayuda de un servo (RC). Esto hace que el proyecto sea versátil y apto para todo tipo de cámaras de fotos. El autor intentó acceder a los hilos del botón de disparo de una cámara digital compacta para poder controlar la cámara electrónicamente. Sin embargo, la miniaturización de los componentes hace que sea casi imposible hacerlo sin destruir la cámara.

Figura 1. Prototipo, construido con algunos componentes de Mecano.

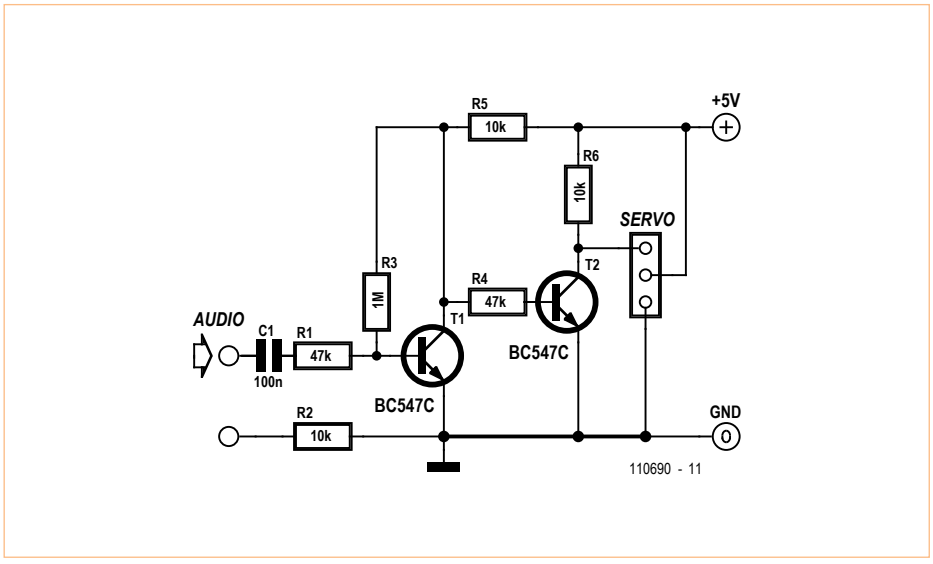


Figura 2. Con este circuito se amplifica la señal de salida de audio hasta el nivel TTL 5 V. Un pequeño adaptador de 5 V de red es suficiente.

Como muestra la **figura 1**, el servo controla una palanca que pulsa el botón de disparo a través de una clavija. También se pueden idear otras soluciones mecánicas para controlar el botón de disparo con un servo.

No es fácil utilizar el puerto USB de una tableta Android en este momento. Esto significaría que para cada tipo de tableta se necesitarían otras instrucciones y software para que el proyecto funcionase. Por eso hemos optado por utilizar el puerto de audio para controlar el servo. Ya que la posición de un servo se determina por un pulso de entre 1 y 2 ms, ó con otros servos entre 0,5 y 1,75 ms. Este pulso se repite cada 20 ms. Con el controlador de audio de la tableta se puede generar directamente dicho pulso. El ancho de pulso de la salida de audio está típicamente por debajo de 1 V y se ha de convertir aún a nivel TTL. Esto se hace con el circuito de la **figura 2**. T1 y T2 amplifican la señal. Como no está garantizado que el 'menos' de la salida de audio sea realmente el menos (y no sea flotante), hay que acoplar el circuito de forma semi DC (R1 y C1). Por lo demás, los valores de todos los componentes no son críticos.

Si no hay que adaptar nada más en el software, hay que instalar el programa copiando el fichero *timeLapse.apk* (en el directorio bin del fichero ZIP) a la tableta a través de USB o una memoria flash. El envío del fichero a la tableta no funciona por correo electrónico. Para instalar el programa hay que utilizar el programa de instalación que viene con la tableta. Se ha de activar "allow installation of non-market applications" en 'settings/application settings/unknown sources'.

El control se puede hacer de dos maneras: a través de la GUI de la tableta (ver **figura 3**) o externo a través de un interfaz Web (ver **figura 4**). En la GUI puedes configurar la cantidad de fotos y el retardo (en segundos, mínimo 3 segundos) en los campos de texto. Con tres barras deslizantes puedes configurar el estado del servo en tres posiciones. La primera es con la cámara en reposo. La segunda es cuando se pulsa el botón hacía la mitad para el enfoque y la tercera posición es para la toma de la foto. Configura bien las barras deslizantes desde el principio, porque el valor varía entre medio y dos ms. Esto está fuera del alcance de algunos servos y pueden dañarse si se suministran estos pulsos por un tiempo prolongado.

Con el botón 'phase' puedes configurar un pulso positivo o negativo. Dependiendo de la tableta puede ser necesario invertir el pulso (esto no es relevante para el audio). Finalmente está el botón start para iniciar y parar la time-lapse.

La aplicación se puede controlar también a través de la Web. La aplicación escucha en el puerto 8090 en la dirección IP de la tableta. Desde el navegador (de otro ordenador) hay que acceder a la tableta con, por ejemplo: `http://192.168.1.101:8090/`. Entonces se mostrará el estado actual. En el formulario puedes introducir nuevos valores y con los enlaces puedes arrancar y parar la time-lapse. Todavía hay que introducir el logotipo de las páginas Web (parte superior a la izquierda) después de la instalación, colocando un fichero llamado logo.jpg en el directorio /sdcard/webserver_TL de la tableta. Este directorio se creará cuando se ejecute la app por primera vez.

Programación de la GUI

En la edición de junio se describió la base para programar el Android y todo lo que necesitas. El autor utilizó también Eclipse [1] con ADT (Android Development Toolkit) [2].

En Eclipse puedes diseñar gráficamente la GUI (ver **figura 5**). Desde el menú puedes arrastrar los botones, campos de entrada, campos de texto, etc. a la pantalla virtual de la tableta. Esto resulta en un fichero `main.xml` en el cual se describen todos los elementos, inclusive los ID únicos de los elementos.

El programa principal se encuentra en `TimeLapseActivity.java` (creado por Eclipse) por debajo de la clase `TimeLapseActivity`. Esta clase se encarga de la interacción con los botones. Para acceder a un botón hay que vincular un objeto al ID anteriormente dicho, por ejemplo:

Figura 3. La GUI para el control de cámara en la tableta.

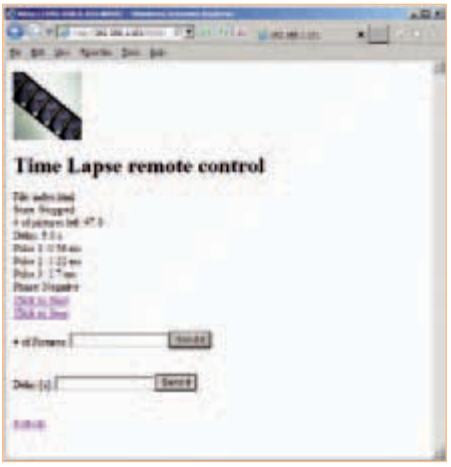


Figura 4. Así es el interfaz Web correspondiente del ordenador.

```
ToggleButton mtoggleButton1 =
(ToggleButton) findViewById(R.
id.toggleButton1);
```

Los botones se pueden utilizar de dos maneras: activa o pasiva. A través de las funciones del objeto se puede leer el

Figura 5. El diseño gráfico en Eclipse.

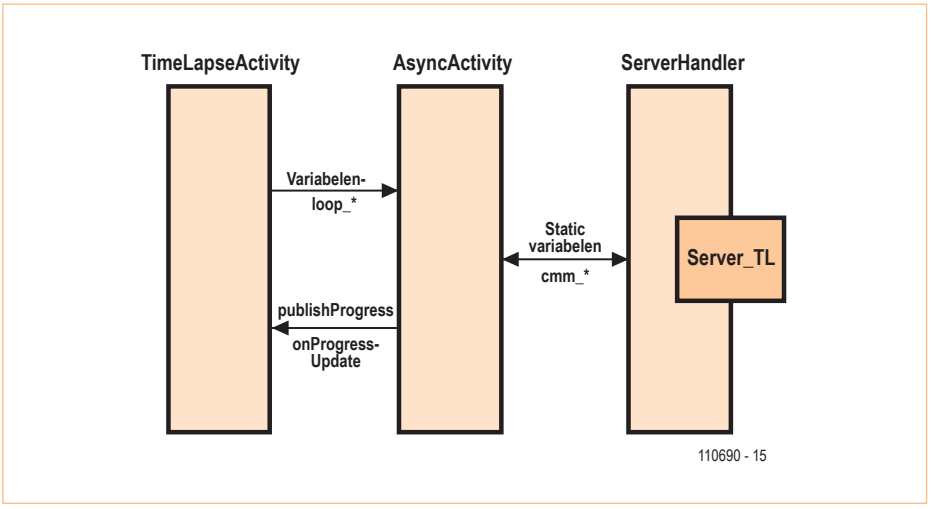


Figura 6. La comunicación entre clases.

estado de un botón y configurar sus valores. Esto es pasivo. Si se tiene que llamar activamente a una clase al pulsar (o cambiar el estado) de un botón, hay que hacerlo con `mtoggleButton1.setOnClickListener(this)`. En este programa sólo se utiliza el botón start de forma activa.

La parte time-lapse de la aplicación es un procedimiento que se ejecuta durante mucho tiempo. Es algo que no puede pasar con la clase principal (`TimeLapseActivity`). Esta clase es responsable de la tramitación de la GUI. La GUI no se refresca, cuando se ejecuta esta clase, sino se congelaría completamente. Además Android termina este tipo de clases si no responden durante un periodo de 5 segundos, porque entonces Android piensa que el programa se ha quedado colgado.

Por eso es por lo que el Android cuenta con la clase `AsyncTask` class. Esta se ejecuta en un thread diferente y puede estar continuamente ocupada. El thread sólo se puede arrancar una vez, no arranca cada vez que arranca una time-lapse, sino que se ejecuta continuamente, también cuando no se ejecuta una time-lapse.

Desde `AsyncTask` no se puede escribir directamente sobre elementos de la GUI (esto conduciría directamente a un cuelgue). Para eso existen las funciones `publishProgress` y `onProgressUpdate`. Se utiliza `publishProgress` para llamar `onProgressUpdate` desde `AsyncTask`. Sólo se puede escribir a la GUI desde la penúltima función. Como sólo hay una única función para controlar todos los botones, se ha definido una determinada acción dependiendo del valor del primer parámetro que se pasa a la función `onProgressUpdate` (ver el código en `onProgressUpdate` y las variables `pp_*`).

La 'comunicación' para arrancar y parar la time-lapse (en la dirección contraria a `publishProgress`) se hace con variables (`loop_state`, `loop_count` y `loop_delay`). Ver la **figura 6** para la comunicación entre las diferentes clases.

La forma de onda de los pulsos del servo se guarda en un array y la función `audioTrack` reproduce lo que está guardado en este array.

Servidor Web

Este proyecto utiliza también una conexión WiFi. Desde una página Web se puede ver el estado y la configuración y se puede controlar la aplicación. Existen tres opciones, groso modo, que hacen que esto sea posible (en términos generales).

Se puede trabajar con un servidor Web suelto de app (hay muchos que se pueden descargar gratuitamente). El servidor muestra ficheros desde el sistema de archivos y es una solución fácil. La desventaja es que con este sistema sólo puedes mirar, pero no controlar. Además hay que renovar continuamente el fichero html, por lo que la memoria flash se deteriora con el tiempo.

También puedes utilizar un amplio paquete como es i-jetty [3] en el otro lado del espectro. Con esto se puede implementar un servidor polifacético, pero esta extensión (con la complejidad correspondiente) no es siempre necesaria.

El autor optó aquí por utilizar el software libre 'android-webserver' [4], incluyendo el código en la aplicación y adaptándolo dónde fuera necesario. La adaptación más importante es que el servidor Web puede seguir leyendo ficheros desde el sistema de archivos (de modo que se puede seguir utilizando imágenes, como en este caso logo.jpg, o ficheros CSS en el directorio `webserver_TL`), excepto uno con un nombre configurable (aquí index.html), desde donde el programa genera una página web dinámica. Como el servidor Web se ejecuta en un thread diferente y no dispone de un acceso directo al programa principal `TimeLapseActivity`, la configuración y estado se transmite a través de variables *estáticas* (ver las variables cmm_*). Se puede ver una variable estática como un sitio fijo en memoria. Esta variable es la misma para

todos los objetos de una clase (contrariamente a las variables normales, donde hay que crear variables nuevas para cada objeto de una clase) y también puede ser leída y escrita desde otra clase. Por eso las variables estáticas se llaman con un nombre de clase y no con el nombre del objeto.
El control de la aplicación se hace mediante análisis del link. Las variables se colocan después de un ? igual que en los formularios HTML (en modo GET). En este caso el URI `index.html?act=start` inicia la TimeLapse. También se configura así el número de fotos y el retardo con, por ejemplo, `index.html?del=5`. Esto se genera automáticamente con la orden HTML FORM. La comunicación de esta configuración y las órdenes van también a través de variables estáticas. Estas variables se leen en el bucle de `AsyncTask`. Otra opción podría haber sido que esta comunicación utilizase un handler, donde habría dos opciones, utilizar un handler común o un handler que provees en el momento de la creación del objeto. Pero este caso no fue necesario.

El resto de las funciones

Un par de observaciones comunes sobre la programación de Android.
Es fácil para el usuario guardar la configuración en el programa (longitud de pulsos, fase, número de fotos y retardo), para que al arrancar la próxima vez ya no haya que introducirla de nuevo. Para eso Android cuenta con funciones especiales, comparable con el registro en MS Windows. Las funciones son parte de la clase `SharedPreferences`. Los parámetros se leen al arrancar el programa. La configuración se guarda al terminar el programa (en onStop). Esta configuración también se guarda cuando se instala una nueva versión del programa.
En las versiones Android actuales no existe un gestor de aplicación. No se puede ver que aplicaciones están ejecutándose (independientemente de si se ejecutan en background). Por eso puede suceder fácilmente que pierdas la aplicación. Si vuelves a ejecutar la aplicación, se inicia una segunda aplicación en paralelo con la primera. Con la aplicación descrita en este artículo esto es un problema, porque los servidores web de ambas aplicaciones escuchan al mismo puerto. Esto hace que el resultado sea imprevisible. Para eso se han incorporado tres medidas en el programa. Primero se coloca el icono en la barra de estado, de modo que puedas volver fácilmente a la aplicación a través de esta barra. Las funciones de la clase `NotificationManager class` sirven para eso. En segundo lugar se indica en el fichero `TimeLapse manifest.xml` que no se puede ejecutar otra versión de este programa en paralelo, con la configuración `android:launchMode = "singleInstance"`. La tableta tampoco puede ejecutar el salvapantallas (esto hace que la aplicación se vuelva inestable por alguna razón desconocida). Esto se hace con la configuración `android:keepScreenOn = "true"`. Por cierto, hay rumores de que habrá menos problemas a partir de la versión 4 de Android.
Durante la instalación de Android, el usuario tiene que dar permiso a la aplicación de forma que ésta tenga determinados derechos para comunicarse con los diferentes componentes hardware y software. En este programa son el acceso al interfaz WiFi, acceso a la memoria flash (para los ficheros del servidor Web) y acceso para bloquear el salvapantallas. Estos permisos se guardan como `uses-permissions` (una errata frecuente es user-permission) en el fichero `TimeLapsemanifest.xml`.
Finalmente puedes adornar el programa con iconos propios. Para eso hay que colocar tres imágenes en formato gráfico .png con las dimensiones 72x72, 48x48 y 36x36 pixeles respectivamente en el directorio res del proyecto en drawable-hdpi, drawable-mdpi y drawable-ldpi.
Todos los ficheros de este proyecto se pueden descargar gratuitamente en www.elektor.es/110690.

(110690)

Enlaces Web

[1] http://eclipse.org/
[2] http://developer.android.com/sdk/eclipse-adt.html
[3] http://code.google.com/p/i-jetty/
[4] http://code.google.com/p/android-webserver/

Android como entorno de desarrollo

Tabletas baratas en electrónica embebida

Elbert Jan van Veldhuizen (Holanda)

Como diseñadores de electrónica estamos acostumbrados a tener que diseñar nuestros propios sistemas o a utilizar una placa de desarrollo. ¿Pero has pensado alguna vez en utilizar una tableta para el desarrollo?

Últimamente se pueden encontrar tabletas con el sistema operativo Android por debajo de 100 euros. Estas tabletas están llenas de electrónica y tienen además un bonito aspecto. Las funcionalidades que ofrece una tableta como estas, son también funcionalidades, por lo general, relativamente difíciles de implementar en un entorno embebido. En este artículo, puedes leer cómo adaptar las tabletas a los proyectos de electrónica.

Hardware

¿Qué puedes esperar de una tableta de menos de 100 euros? Para empezar, una caja bien acabada. Sin ningún reparo, puedes dejar una tableta en el salón, montarla en el salpicadero del coche o utilizarla como cubierta frontal de una caja para conseguir al instante un panel de mandos con una apariencia profesional.

Y con esto llegamos a una utilización muy importante. Una tableta tiene un display con retroiluminación y un touchpad, con lo que es perfectamente apta para formar un interfaz gráfico de usuario (GUI). A través de la combinación de una pantalla y un touchpad, se crean botones (virtuales) (que incluso pueden estar relacionados con el contexto) y campos de entrada e incluso se puede presentar una estructura completa de menú. Esto nos lleva a pensar que las tabletas también son aptas para aplicaciones multimedia. Así que, con ella se puede ver incluso imágenes en vivo de una cámara (IP) o videos de instrucción. Los displays de las tabletas más baratas tienen una diagonal que varía desde 7” (18 cm) hasta 10” (25 cm) y resoluciones desde 800×480 hasta 1024×600 pixeles. La pantalla táctil es generalmente del tipo resistivo (sobre la que sólo puedes pulsar con un dedo a la vez). Normalmente, esto es suficiente para la mayoría de nuestras aplicaciones.

Puede que ahora tengas la impresión de que una tableta no es más que un terminal ‘tonto’. Nada más lejos de la realidad. Incluso las tabletas más baratas tienen un procesador ARM11 con más de 600 MIPS en potencia de cálculo. Esto es entre 10 a 100 veces más rápido que los microcontroladores que se utilizan normalmente en los diseños. Los programas que requieren mucha potencia de cálculo es mejor ejecutarlos en la tableta. Puedes pensar en los cálculos que normalmente se ejecutan en un DSP, como son las transformaciones de Fourier y los filtros. Sin embargo, el sistema operativo Android no es un sistema operativo en tiempo real, de modo que una tableta no se puede utilizar para cálculos de este tipo. Android sí es lo suficientemente preciso como para funcionar en tiempo real en una escala de segundos (tiempo real aproximado).

En la edición de junio de este año se describió ampliamente cómo programar apps en el Android [1]. El sistema operativo Android es multitarea. Por eso pueden ejecutarse al mismo tiempo varias apps de varias funciones.

Otra función importante de una tableta es la conexión a Internet. Todas las tabletas disponen de una conexión WiFi para conectarse. Para el Android existen varias apps: por ejemplo, un servidor Web, un servidor ftp y varias aplicaciones de correo electrónico. También estas apps pueden ejecutarse al mismo tiempo (en el background). Puedes utilizar fácilmente la conexión a Internet para ver datos a distancia, descargarlos o controlar electrónica. Algunas tabletas (más caras) disponen también de un interfaz móvil 3G. Con esto la aplicación ya no está limitada a un entorno con un router WiFi, sino que puedes establecer una conexión a Internet desde casi cualquier sitio.

Por lo demás, las tabletas disponen de memoria (flash). La mayoría de ellas disponen de una ranura micro-SD para la memoria adicional. Con esto puedes equiparla (en el momento de la publicación) de una capacidad de 32 GB. Una utilización típica de esto es una aplicación de registro de datos. Con tal capacidad puedes guardar, por ejemplo, un flujo de datos de mil muestras por segundo durante un año. Esta capa-

cidad también puede servir para las aplicaciones multimedia anteriormente dichas.
Aparte de eso, la mayoría de las tabletas dispone de altavoces incorporados (con salida externa), un micrófono y una cámara Web. La tableta proporciona también (por el puerto USB) una tensión de 5 V estabilizada de la batería que puede estar incorporada o no. Aunque esta no sea la razón por la que integrar una tableta en la electrónica, es bienvenida.
De momento sólo hemos hablado de tabletas. También existen teléfonos Android por debajo de 150 euros. Tienen una pantalla mucho más pequeña, pero vienen estándar con la funcionalidad 3G. Si la funcionalidad 3G es más importante que el tamaño de la GUI, entonces un teléfono Android resulta muy indicado. Por lo demás, estos teléfonos disponen también de Bluetooth como interfaz de comunicación. El proyecto Amarino [2] lo utiliza para interconectar una placa Arduino con un teléfono Android.

Interfaz USB

Todas las tabletas tienen un puerto USB. Este parece ser el medio correcto de interconectar electrónica externa con la tableta. En teoría es una tarea fácil, pero en la práctica cuesta bastantes quebraderos de cabeza. Para empezar tenemos que darnos cuenta de que las tabletas Android surgieron de los teléfonos Android. Los teléfonos tienen un puerto USB que funciona como esclavo al conectarlo, por ejemplo, a un PC. Por lo tanto, el puerto USB de una tableta Android también es del tipo esclavo. Esto significa que la electrónica conectada tiene que ser del maestro. Normalmente se requieren controladores complejos para esto. Sin embargo, muchas tabletas pueden conmutar sus puertos USB al modo maestro (donde se requiere un cable especial de conversión) o tienen un segundo puerto que funciona como maestro. El objetivo que se persigue es poder conectar un ratón, un teclado o una tarjeta de memoria. También se pueden conectar los controladores más sencillos (con puerto USB esclavo). Microchip tiene disponible un código que permite comunicar con un aparato Android configurado como esclavo. También existen varias placas de desarrollo [3].

Figura 1. Conector USB micro-B a bus USB cable-A para utilizar una tableta en modo maestro.

Aparte de hardware también se necesitan dispositivos de control correctos. Y aquí empieza a ser menos fácil. No existe un soporte estándar en Android para aplicaciones USB diferentes a las descritas anteriormente. La última versión (en este momento) de Android (2.3.4 y 3.1) tiene más posibilidades, pero no son directamente aplicables. Para poder instalar el dispositivo de control correcto, hay que instalar una versión adaptada de Android. Para eso se necesita el software (la denominada 'ROM'), pero también el acceso a root en la tableta. Existen apps que pueden proporcionar este acceso a root en las tabletas más baratas con una instalación de Android estándar. No obstante, la generación de una 'ROM' nueva es tarea para usuarios avanzados, donde también es importante disponer de todos los dispositivos de control. Esto es mucho trabajo para un solo proyecto. Sin embargo, es digno de consideración si se utiliza una tableta determinada en los proyectos.
Afortunadamente parece que este problema se va a solucionar. Google, el creador de Android, trabaja ahora en la versión 4 de Android, llamado *Ice Cream Sandwich*, que une la versión 2 (para aparatos móviles, incluso las tabletas baratas) y la versión 3 (para tabletas). Google tiene precisamente como objetivo poder interconectar los dispositivos Android con muchos aparatos externos. Para eso se ha desarrollado especialmente la clase software 'accessory' [4] (ciertamente existe esta funcionalidad en las versiones 2.3.4 y 3.1 de Android, pero no existe un soporte estándar en la ROM). El puerto USB está configurado como maestro, de modo que se pueden conectar controladores sencillos. Esto simplifica la conexión de aparatos USB a las tabletas. Probablemente se le provea a la ROM de soporte estándar para eso, de modo que ya no hagan falta actualizaciones.

A finales de 2011 se espera la versión 4 de Android. Los requisitos del sistema de la nueva versión son mayores que los que ofrecen las tabletas actuales. Probablemente tardará un poco hasta que haya tabletas baratas con esta nueva versión, porque la potencia del procesador que se necesita tiene que bajar primero de precio.

Arquitectura

Si se utiliza una tableta en un proyecto de electrónica, el sitio dónde programar la inteligencia (tableta o electrónica) es una elección arquitectónica importante. La ventaja de programar la inteligencia en la tableta es que hay un claro entorno de desarrollo, con un API que proporciona acceso directo a los diferentes componentes de la tableta. La tableta dispone además de una capacidad grande de procesamiento y de memoria. Sin embargo, no son posibles las aplicaciones en tiempo real. La elección de programar la inteligencia con el controlador conectado se hace principalmente si

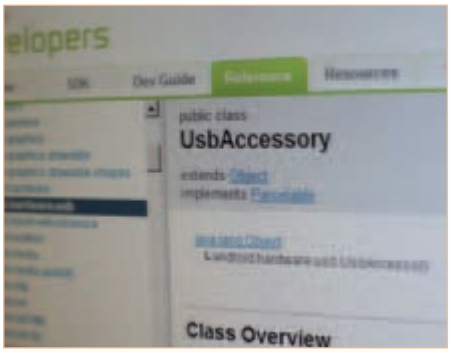


Figura 2. El API de Android está bien documentado.

se requiere un control de la electrónica en tiempo real (y/o a alta velocidad).
En la práctica una solución hibrida será evidente, donde la app de la tableta se ocupe de la funciones cercanas a la periférica de la tableta, mientras el controlador conectado se encargue del control directo de las E/S.

Aplicaciones

Para hacernos una idea de lo que se puede hacer presentamos aquí varias aplicaciones con la arquitectura que precisan:

- Registrador de datos, donde se introduzca la configuración a través de la GUI. Se pueden mostrar los datos en forma de gráfica en la GUI o a través de una página Web. Los datos en bruto se pueden descargar a través de la Web o ftp. Para eso el hardware que se requiere es un ADC conectado al puerto USB. El software requerido son tres apps: un servidor Web y un servidor ftp (para eso se pueden encontrar apps estándares) y una app escrito por uno mismo para leer los datos y guardarlos, más una pantalla para las configuraciones y representación de la gráfica.
- Telemetría de un coche (de carreras), donde se muestre la información actual en la GUI, se pueda mostrar en tiempo real a través de una conexión 3G y se guarde en una tarjeta de memoria. Para eso se requiere un interfaz USB ODB2 y el software para el interfaz, la configuración, lectura y almacenamiento de los datos. Si sólo hay que leer los datos por cada vuelta, bastaría una app de un servidor ftp estándar. Si se lee en tiempo real (aproximado), entonces uno mismo tendrá que escribir una app propio a base de, por ejemplo, una comunicación serie.
- Domótica: controlar un panel solar térmico y una caldera, utilizando la previsión del tiempo en Internet, controlándolo a distancia a través de un sitio Web y una GUI en la tableta. Aquí se necesita la misma configuración que la del registrador de datos. Adicionalmente se debe poder tomar decisiones como utilizar el panel solar térmico a base de la información disponible en Internet, por ejemplo, la cantidad de sol y la temperatura exterior prevista. Para eso hay que escribir una app especial que, por ejemplo, lea los RSS y destile la información relevante.
- Osciloscopio digital, donde se pueda adaptar la configuración en la GUI y enviar las gráficas por correo electrónico si fuera necesario. También aquí se requiere la configuración descrita para el registrador de datos. Una gran parte de la inteligencia está en la electrónica conectada, debido a la indispensable velocidad de las señales.
- "News ticker", con una tableta como interfaz y servidor web (para el control remoto). También aquí se requiere una extensa electrónica debido a la gran cantidad de E/S y la velocidad del multiplexado del panel del "News ticker".
- Mando de control de una maquina de café, con correo electrónico al servicio de mantenimiento en caso de fallo o para reponer el café. La tableta puede funcionar aquí como el corazón del sistema, pero para el control y lectura de todos los componentes se requiere una electrónica de E/S.
- Consola estándar de mantenimiento. Una tableta puede servir perfectamente como consola externa de fallos, de configuración y de mantenimiento. En su forma más simple puede comunicarse directamente con la electrónica a través del interfaz USB y un terminal en la tableta. Así puedes ver, por ejemplo, mensajes de arranque o puedes mostrar los comandos del terminal. Una variante más compleja es una consola conectada con un convertidor USB JTAG. Visto los bajos costes, el cliente debería tener una consola así guardada en un armario.
- Prototipado. Como es fácil escribir una app, una tableta puede hacer que el desarrollo de prototipos y prueba de concepto sean rápidos y eficientes. Debido a la gran cantidad de funciones estándar en Android y las apps disponibles, es posible utilizar muchas funcionalidades en poco tiempo.

Figura 3. Aplicación de una tableta en domótica.

Conclusión

Las tabletas baratas pueden ser un componente efectivo, barato y rápido para implementar un proyecto de electrónica. Tienen un diseño bonito y mucha funcionalidad de fácil acceso. En este momento conectar una tableta con electrónica externa a través del puerto USB requiere bastante trabajo. Esto será más fácil el año próximo por la llegada del Android 4, con la que se abre una vía para muchas aplicaciones.

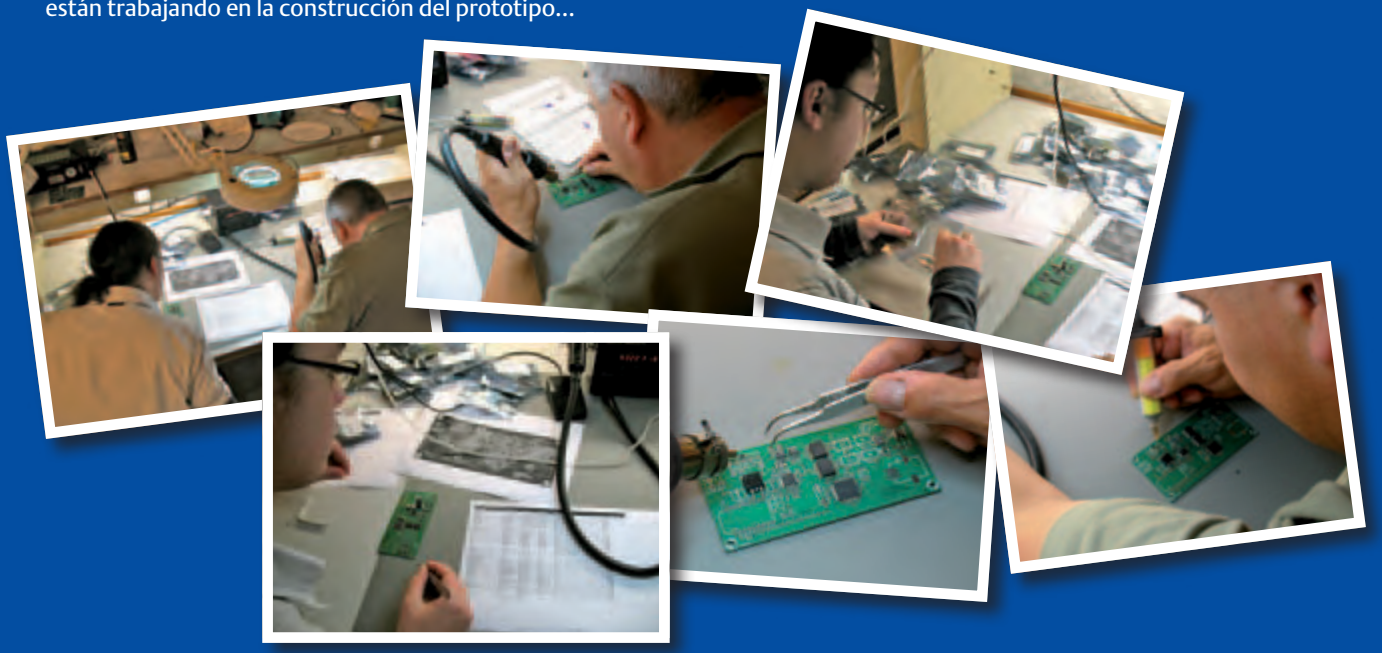
(110667)

Enlaces Web:

[1] www.elektor.es/110265
[2] www.amarino-toolkit.net
[3] www.microchip.com/android
[4] http://developer.android.com/guide/topics/usb/accessory.html

Trabajos en ejecución

Normalmente el laboratorio de Elektor es una fortaleza cerrada no accesible a los lectores (a menos que visites el castillo durante la cita anual del día nacional de los monumentos...). Por lo general todo lo que ocurre en el laboratorio es a puerta cerrada. Esto no quiere decir que no haya nada interesante. He aquí un pequeño 'peepshow' de dos proyectos en los cuales está trabajando el laboratorio en este momento. El primero es un nuevo medidor LCR. El diseñador Raymond Vermeulen y su compañero Jan Visser están trabajando en la construcción del prototipo...



La segunda sesión de fotos muestra los esfuerzos del diseñador (principal) Chris Vossen, trabajando en la impresora 3D. La fecha de publicación es aún secreta, pero puedes ver más información sobre la impresora 3D y el último estado en que se encuentra en www.techthefuture.com/3D-printer.

(110527)

Exorcismo de LED

Parpadeo fantasmagórico de LEDs (2)

Dr. Thomas Scherer (Alemania) & los lectores de Elektor

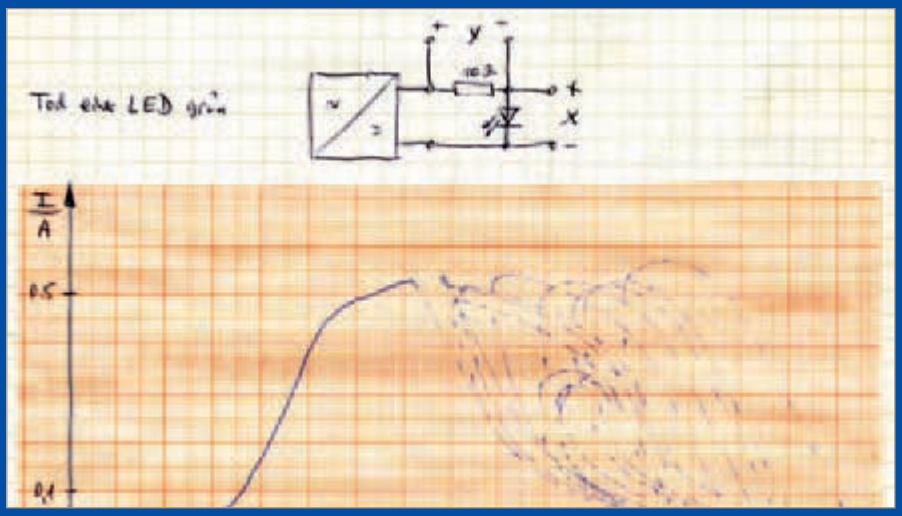


Figura 1. Trazado de curva característica original de un LED agonizante de los años 70.

En la edición de septiembre publicamos mi asombro sobre LEDs que parpadeaban y no debían hacerlo, así como una invocación para que el espíritu residente en tales LEDs se manifestara. Recién impresos y enviados los primeros ejemplares, ya teníamos algunas de las primeras interesantes, inteligentes y sorprendentes reacciones de los lectores de Elektor. Pero primero, hagamos una escéptica descripción de los hechos:
Un LED indicador de carga en un sacacorchos eléctrico y un LED de potencia de 1 W en una lámpara decorativa empezaron a parpadear sin más, pasando posteriormente a mejor vida. Ya que el circuito sólo consistía en un LED con una resistencia en serie para limitar la corriente, como ingeniero electrónico, uno se frota los ojos completamente sorprendido. Después, la editorial decidió apelar al conocimiento de los lectores de Elektor para investigar el misterio. Y así explican el fenómeno los lectores:

Wolfgang Bredow de Lilienthal escribe:
"Me recordó inmediatamente a un experimento que hice a finales de los 70. Por aquel entonces, registraba las curvas características de todos los componentes posibles gracias a un trazador de curvas. Una de las veces registré la muerte intencionada de un LED. Incluso tuve que levantar la aguja (ver la ***figura 1****).*
Como puede verse, la víctima (un LED verde) fue llevada más allá de su punto de trabajo. Con aproximadamente 7 V y 500 mA, lo primero que hice fue cambiar su color a rojo oscuro. Después el LED empezó a parpadear (¡¡¡ajá!!!), lo cual llevó a la aguja a realizar movimientos alocados. Las líneas entrecortadas son el resultado de una mecánica demasiado lenta, incapaz de seguir los continuos encendidos y apagados de la corriente del LED."
Por cierto, el señor Bredow asegura que este ha sido su único asesinato intencionado en cuanto a componentes electrónicos se refiere...

El Dr. Ing. Ulrich Pilz de Hohen Neuendorf:
"Yo también he observado el parpadeo fantasmagórico justo antes de que un LED falle. Mi explicación es que al dilatarse y contraerse la estructura del LED con los encendidos y apagados (tensiones térmico-mecánicas en el material) se deteriora el contacto con el chip del LED. Esto conduce a –en principio reversibles– interrupciones en el flujo de corriente. Otras veces, el circuito se abre permanentemente y el LED (y no el chip de éste) quedará inservible."
El Sr. Pilz no fue el único en afirmar que el contacto entre los cables de las patillas y el chip fuese el culpable del parpadeo. A un lector incluso le inspiró la idea para un circuito de "devolver LEDs a la vida"...

Karl-Heinz Ziener de Lauenstein:
"Supongo que en los LEDs aparecen roturas del tamaño de un pelo. Probablemente los cables de conexión se rompan por la dilatación térmica. En mi opinión, el parpadeo se da debido a que los LEDs se conectan en frío. El LED ilumina y se calienta, mientras tanto el material se expande en la rotura y se separa el terminal, cortando la corriente. Sin corriente no hay calor, con lo que el material se contrae nuevamente y el contacto vuelve a su posición original. Y así otra vez.
Una de las posibles soluciones (no del todo fiables) se muestra en la ***figura 2****: se conecta un condensador en paralelo con el LED. Cuando el LED se apaga, el condensador se carga a la tensión de la batería. Si el LED, frío internamente, quiere volver a iluminar, la energía acu-*

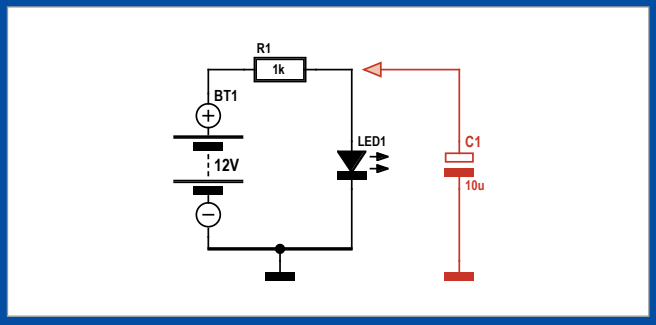


Figura 2. Para su reparación: la carga de un condensador debería soldar de nuevo los contactos en el chip del LED.

mulada en el condensador se dirige al punto de rotura y lo suelda de nuevo. Éxito en la reparación."
Esto sólo resulta satisfactorio naturalmente si el LED aún parpadea y no está muerto del todo ;-)

Gwyn Evans envió desde su smartphone:
"La razón obvia de estas apariciones es que la pila de 9 V probablemente no esté dando una corriente continua en absoluto, sino corriente alterna entrecortada, sin duda debida a colapsos cercanos entre sí, en el continuo espacio-tiempo con un ritmo de 50 Hz."
Bien, ya empezaba a asustarme. Pero después vino esto: "Es posible alargar la vida útil de un LED si lo hacemos funcionar durante largo tiempo con pulsos a 1 kHz."
Después de esto, ¿debería preguntarle a las compañías eléctricas porque no pueden conectar la frecuencia de los generadores de las turbinas eólicas directamente a la red?

Hubert Maiwald de Neutraubling:
"La mayoría de LEDs tienen un contacto en la parte superior y éste va pegado a la estructura con cinta adhesiva plateada, que también sirve como conexión eléctrica.
Esta cinta adhesiva puede deteriorarse debido a un descuido, como una soldadura demasiado caliente, sobretemperaturas, mala disipación, etc. Primero se forman pequeñas burbujas entre el LED y el área de contacto.
Esto empeora debido a la conductividad térmica entre los contactos es cada vez más baja, disipando el calor con mayor dificultad. Posteriormente se forman más y más burbujas. La cosa sigue yendo a peor. A temperatura creciente, las longitudes de onda se desplazan al rango de la baja frecuencia.
Finalmente el contacto es casi inexistente, sumado a las dilataciones de los distintos materiales, en fin, la parte superior del LED está bastante más caliente que la cara inferior. Al dejar de circular corriente, todo se enfriará y volverá a su posición original. Así hasta que haga otra vez contacto eléctrico y el cristal del LED se caliente de nuevo...
La constante de tiempo térmica de esta oscilación está en un rango entre 0,1 y 10 s, y depende de la corriente y la potencia disipada. El funcionamiento de este parpadeo es muy similar al de un viejo mecanismo de relé bimetálico. Este fenómeno no es demasiado común, ya que el contacto no se daña por un precalentamiento si no estaba dañado el adhesivo previamente o la conexión abierta o defectuosa. Generalmente se trata únicamente del adhesivo, y quedará abierto de forma irreversible."
Suena lógico, ¿no?

Ahora, queridos lectores de Elektor, ¿cual creéis que es la explicación más fiable? Que se trata de algo térmico, podemos asegurarlo casi por completo. En la electrónica siempre hay puzzles nuevos, y como en este caso, en situaciones que ni siquiera nos imaginamos...

(110668)

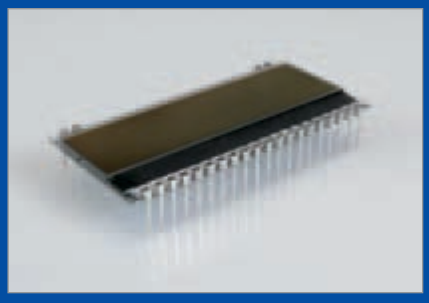
Terminales a la medida correcta

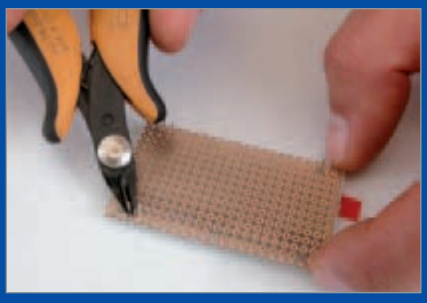
Thijs Beckers (Redacción NL)

En un Labcenter anterior tratamos la colocación de una placa de pruebas entre un LCD y la placa (madre) con el objetivo de poder retirar a menudo el display sin romperlo. Aquí nos referimos a los displays frágiles como son los de la serie DOGM de Electronic Assemblies. Ahora hay también una segunda utilidad para esta placa de pruebas.
Los displays DOGM anteriormente mencionados tienen patas bastante largas (sus terminales son más bien largos). Esta altura es necesaria para poder equiparlos de backlight que – como su nombre indica – se coloca detrás del display. Sin embargo, si se anula esta iluminación, es mejor que el display esté más próximo a la placa, de modo que el conjunto se pueda instalar en una caja de forma más compacta. La última opción que queda para bajar la altura del display, en caso de utilizar un zócalo para montar el display, es el acortamiento de los terminales. Luego lo más práctico es dar la misma longitud a todos ellos. He aquí de nuevo nuestra placa de pruebas: coloca el display con los terminales atravesando la placa al máximo y corta la parte que sobresale. La longitud que queda es perfecta para colocarlo en el zócalo. Al tener todos los terminales la misma longitud, el display no se tambalea en el zócalo.

¿Tienes algún consejo práctico para nosotros? Envíanoslo por correo electrónico a la dirección: redacción@elektor.es

(110664)





Itsy Bitsy Spider...

Raymond Vermeulen (Laboratorio de Elektor)

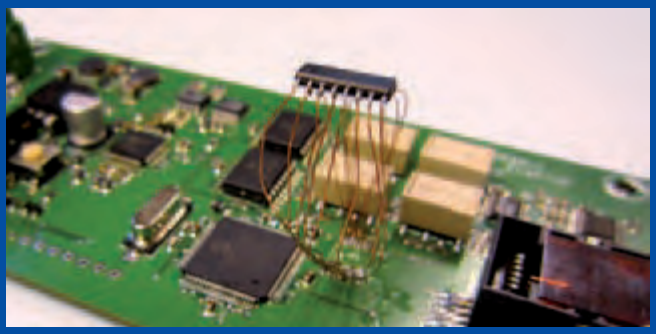
Después de haber trabajado duro en un proyecto llega el momento esperado: encargas la placa, los materiales y cuando ha llegado todo, empiezas con la construcción. Entonces resulta que la 'footprint' (huella) de uno de los integrados no coincide. ¿Y ahora qué?

En mi caso se trataba de un driver de relé DS2003. Aparentemente, ya no se fabrica la versión con el encapsulado TSSOP desde el 2009. Sólo se fabrica la versión SOIC. Está bien saberlo, ¡pero demasiado tarde!

En lugar de pedir una placa nueva con la footprint corregida, lo que llevaría un par de semanas (se podría hacer más rápido, pero eso encarecería mucho el coste) y sería una pena por todo el tiempo y esfuerzo dedicado a la parte ya construida de la placa SMD, en lugar de eso, empiezas a improvisar.

La solución en este caso (ver foto) es 'rápida' y 'muy sucia'. Afortunadamente este integrado se utiliza sólo para conmutar varios relés. Este método no es muy útil en aplicaciones donde la pureza de la señal sea importante.

Qué es lo que hice: Primero soldé un trozo de hilo lacado a cada terminal del integrado. Después corté todos los hilos a la misma longitud y doblé los extremos. La razón por la cual los



hilos son tan largos, no es sólo para permitir que puedas pasar entre medias con la punta del soldador sino también para que el hilo no se desprenda de un lado por la conducción del calor mientras estas soldando el otro. La soldadura no es muy difícil si tienes mano firme. Es difícil para gente con manos menos firmes, como tengo yo, pero al final se consigue. Te recomiendo fijar primero el integrado y liberar los hilos de la laca en los puntos de soldadura. Pero lo mejor es siempre, la utilización de la footprint correcta en la placa...

(110692)

Bus maloliente

Thijs Beckers (Redacción NL)

Durante la redacción del artículo sobre el uso de tabletas baratas en electrónica embebida (ver en otra parte de ésta edición) noté de repente el olor típico a electrónica quemada. Siguiendo mi nariz (como miembro del equipo de evacuación siempre estás más o menos obligado a responder a este tipo de olores) llegué al despacho de la redacción alemana más allá, donde mi colega Jens Nickel trabajaba en el proyecto Elektor Bus. Trabajaba, así resultó ser, ya que el condensador electrolítico de tantalo del convertidor USB/RS485 que formaba parte del sistema del Elektor Bus [1], se había quemado (ver foto). Jens ya se encontraba en el laboratorio con la placa defectuosa, pero el olor a baquelita quemada aún perduraba.

No estábamos muy contentos con lo que había pasado. ¿Qué hacer ahora si esto puede pasar con todas las placas vendidas? ¿Se trataba de un error de producción o estaban todos los condensadores electrolíticos de cada placa montados al revés? ¿Tenemos que pedir que nos devuelvan todas las placas vendidas? ¿Fue un error nuestro o del montador? ¿Quién lo va a pagar? Etcétera.

La verificación del stock en nuestro almacén debería darnos una respuesta definitiva. Ya se había acabado el día, así que hubo que esperar hasta el día siguiente, ya que nuestro almacén está a unos diez kilómetros del castillo en un polígono industrial.

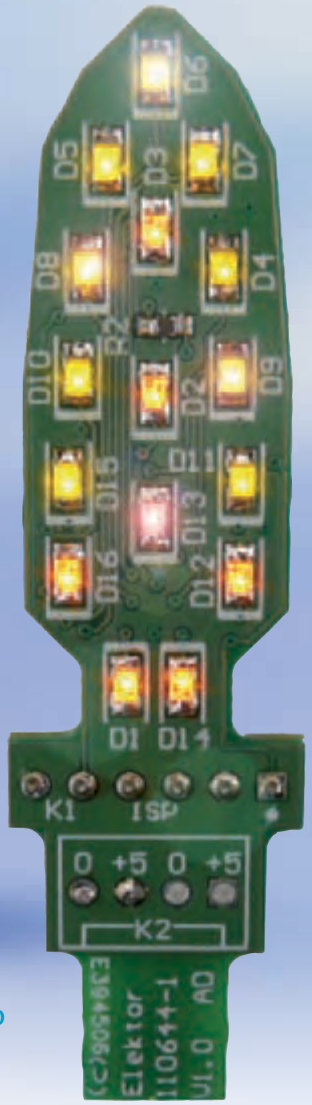
En fin, al día siguiente se verificó directamente si el resto del stock (y por lo tanto también las placas ya suministradas) estaba correctamente montado o si teníamos un problema gordo. ¿Cuál fue el resultado? Pudimos respirar tranquilos, ya que todas las placas del stock que habíamos verificado estaban montadas correctamente. Lo que nos llamó la atención fue que también se habían utilizado otros componentes en nuestras placas del almacén (ver foto). Entonces comenzó a surgir una idea... Los módulos que utilizaba Jens eran prototipos montados en el laboratorio, por eso la placa llevaba componentes diferentes a la de la producción en serie. Probablemente la persona que la montó no se había dado cuenta de que en los condensadores de tantalo están marcados el + (en los condensadores electrolíticos normalmente están marcado el -). Finalmente se quedó en agua de borrajas. Así ves que te puedes encontrar con cualquier cosa cuando desarrollas algo. ¡Por lo menos no es aburrido! Lo que nos sorprendió fue que el condensador de tantalo aguantase tanto tiempo en el estado con los polos invertidos. No se notó nada en el funcionamiento del circuito.

(110693)

Enlaces Web:

[1] www.elektor.es/110258

Vela electrónica con LED
Para apagarla, ¡sópla!



En el comercio existen imitaciones de velas con LED como elemento de iluminación. Aquí se describe un montaje sensiblemente diferente con algunas características poco habituales, ya que ¡una vela se apaga soplando sobre ella!

Antoine Deschamps (Francia)

La luz de una vela es, por naturaleza, variable. Así pues, vamos a realizar animaciones para simular la llama de una vela. Habrá una secuencia de encendido, una secuencia de apagado y algunas secuencias animadas destinadas a reproducir, con más o menos fidelidad, los temblores de una llama natural. Para realizar las animaciones necesitamos un microcontrolador. Nuestra elección se decantó sobre el PIC16F1827 de Microchip, que posee una memoria de programa de 4 KB, suficientes para el código de gestión de la aplicación y las definiciones de las frecuencias de las animaciones.

Este microcontrolador también posee funciones de gestión de teclas táctiles y la facilidad de su puesta en funcionamiento, lo que nos ha conducido a disponer, sobre la cara vista de nuestra placa, una zona de cobre que hace la función de tecla táctil. Es esta tecla táctil la que nos permitirá encender la vela.

Para realizar nuestro objetivo de poder apagar la vela soplando sobre ella (un poco insistente), hemos «convertido» un sensor de temperatura a la forma de una resistencia NTC. ¡Y funciona!

El sensor de soplo

Como se ha dicho más arriba, utilizamos una resistencia NTC (R2 en la **Figura 1**) como sensor de soplado. Hemos elegido un modelo con encapsulado SMD de 0603, con una resistencia nominal de 220 Ω a 25 °C, con un coeficiente beta de 3540 K y una potencia máxima de 180 mW. Un coeficiente beta elevado permite una variación de la resistencia más grande en función de variaciones de temperatura.

El principio de nuestro montaje es el siguiente: se considera que el soplido de una persona es sensiblemente similar a la temperatura ambiente del lugar. A priori, soplar sobre una NTC que ya está a la temperatura ambiente, no hará variar su resistencia. El truco está en hacer circular suficiente corriente por la NTC para que se caliente. Así, un soplido un poco marcado hará bajar la temperatura, la cual podremos medir seguidamente, gracias a una entrada analógica del microcontrolador.

Nuestro circuito dispone de un montaje en puente divisor, con una resistencia común de 100 Ω (R1). En el encendido y, con una temperatura ambiente de 25 °C, la corriente es de $I_{25} = 5 / (100 + 220) = 16$ mA. La tensión en los bordes de la NTC es de $U_{25} = 220 \times I_{25} = 3{,}5$ V, y la potencia disipada por la NTC es de $P_{25} = U_{25} \times I_{25} = 3{,}5 \times 16$ mA = 56 mW. Por lo tanto, estamos lejos del máximo de 180 mW. Pero esa potencia es suficiente para provocar un incremento de la temperatura del componente, disminuyendo así su resistencia, aumentando la corriente, etc. Todo el conjunto se estabiliza al cabo de unos segundos, tal y como lo muestran las curvas gráficas (ver **Figura 2**). Nuestro programa espera durante unos 20 s a que la temperatura se haya estabilizado antes de comenzar a tener en cuenta las medidas. Es delicado representar la evolución de esta curva en el momento en que se sopla sobre la NTC. A base de pruebas, hemos llegado a definir que una variación positiva del orden de 25 mV es suficiente para detectar un soplido humano.

Desde el punto de vista del programa, es necesario hacer coincidir el ritmo de adquisiciones analógicas con las velocidades de evolución de las señales medidas. Hemos colocado un primer filtrado para eliminar el ruido, después hemos realizado la media sobre 16 valores sucesivos, de manera que podamos detectar cuándo un valor filtrado se desvía de la media de los 16 valores en más de 25 mV. El resultado ha sido que es necesario soplar razonablemente cerca de la vela para apagarla. Este estado es algo variable ya que el conjunto es bastante sensible a las corrientes

Características técnicas

- Simulación realista de la llama
- Apagado por soplado
- Encendido por tecla táctil
- PIC16F1827 programado en C, código fuente suministrado
- Sensor de soplado basado en resistencia NTC
- Dificultad de ensamblado: mediana

de aire. En este circuito no tenemos necesidad de una gran precisión, ya que analizamos las diferencias de temperatura con la media de las temperaturas registradas. Por ello no ha sido necesario colocar una tensión de referencia sobre la placa ni, incluso, utilizar la referencia interna del microcontrolador. El intervalo de medida analógico queda así delimitado por los 0 V de una parte, y por la tensión de alimentación de la otra.

La tecla táctil

La tecla táctil que permitirá encender la vela está conectada directamente a la entrada analógica 0 del micro IC1 (**Figura 1**). En la documentación del PIC16F1827, el reloj de las teclas táctiles capacitivas (*capacitive sensing module*) está descrito de forma clara y su programación no supone ningún problema. Hay que tener en cuenta un detalle, ya que no tenemos una noción inmediata de la frecuencia de este oscilador que forma parte de la tecla táctil. Recordemos que es la variación de capacidad de esta tecla táctil, al acercar un dedo humano, la que provoca la variación de la frecuencia que debemos medir o, de manera más precisa, comparar con un valor de referencia. La salida del oscilador está conectada a Timer 0, el cual cuenta libremente.

Para poder evaluar estas frecuencias, hemos copiado, sobre la salida conectada en el punto de prueba, el bit de mayor peso del registro de conteo del Timer 0, TMR0. Así, hemos obtenido sobre este punto de prueba, accesible con el osciloscopio, una frecuencia del orden de 130 Hz, que pasa a menos de 90 Hz cuando el dedo toca la zona de contacto. Esta frecuencia sobre el octavo bit del registro corresponde a una frecuencia 256 veces más elevada sobre la entrada del Timer 0, es decir, alrededor de 32 kHz.

Nuestra aplicación está sincronizada con un reloj de 1 kHz (ver más adelante), por lo que es suficiente examinar el contenido del registro TMR0 con cada pulso de 1 ms. TMR0 vale alrededor de 32 cuando la tecla está libre, mientras que este valor desciende a las proximidades de 23 cuando apoyamos el dedo sobre la tecla. Una vez leído, se fuerza el estado del registro TMR0 a 0 para que comience a contar de nuevo. En nuestra aplicación hemos fijado el umbral en 26 que es un valor bastante bajo, por lo que será necesario realmente colocar el dedo sobre la tecla para activarla. Si aumentamos este valor daremos una mayor sensibilidad a la función táctil.

Una puntualización: es importante que, siempre que sea posible durante la puesta a punto del montaje, prever uno o varios puntos de prueba. ¡El osciloscopio también es una herramienta que ayuda en la programación!

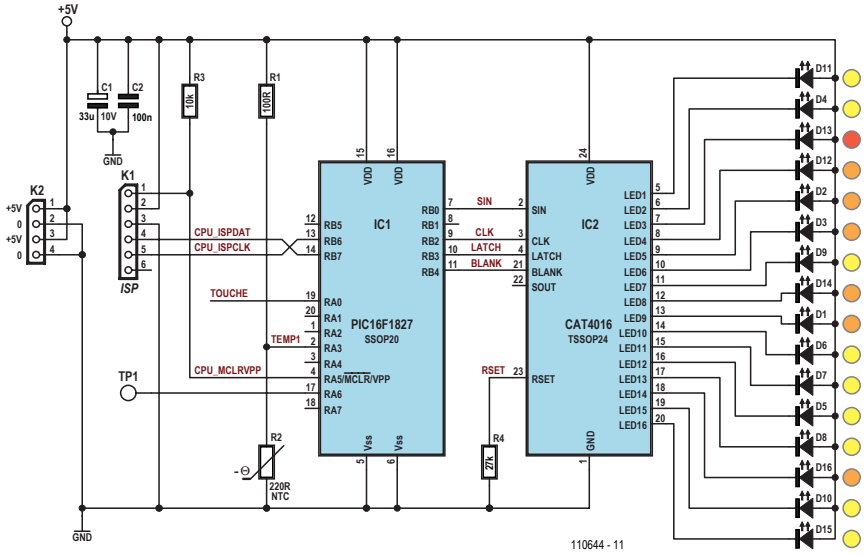


Figura 1. Esquema eléctrico de la vela de LEDs. El número de componentes ha sido reducido al mínimo para poder ser alojado en la superficie ocupada por los LEDs en la cara superior de nuestra placa. El circuito está alimentado con 5 V, con un condensador electrolítico (C1) para estabilizar la tensión. La puesta en servicio del microcontrolador IC1 se ha reducido a su mínima expresión. Usamos el oscilador interno ya que no necesitamos precisión en el funcionamiento.

La llama

Esta aplicación con LED está pensada para decorar, no para iluminar. Así pues, vamos a utilizar LEDs de baja corriente de unos 2 mA como máximo. Esta gama de potencia permite la elección de LEDs SMD, lo que permitirá colocarlos sobre una superficie de circuito impreso relativamente pequeña.

El control de los LEDs que constituyen la llama, 16 en total, ha sido confiado a un circuito especializado, IC2, el CAT4016 de ON Semiconductor. Este componente integra una regulación de corriente constante para 16 LEDs, donde el valor de la corriente viene determinada por una resistencia externa (R4). La corriente que pasa por cada rama de LEDs vale 50 veces la corriente que atraviesa la resistencia conectada sobre el terminal R_{set}, sabiendo que la tensión sobre

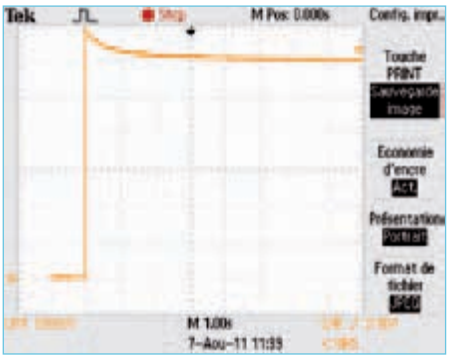


Figura 2. La temperatura del sensor de soplado se estabiliza al cabo de unos segundos.

Lista de componentes

Resistencias (SMD 0603):
R1 = 100 Ω
R2 = CTN 220 Ω, Epcos B57311V2221J60 (p.ej. Farnell 129-9912 o RS Components 706-2702)
R3 = 10 kΩ
R4 = 27 kΩ

Condensadores:
C1 = 33 µF 10 V aluminio electrolítico, encapsulado C, p.ej. Panasonic EEE1AA330SR
C2 = 100 nF, SMD 0603

Semiconductores:
IC1 = PIC16F1827–I/SS (SSOP20), Microchip
IC2 = CAT4016Y-T2 (TSSOP24), ON Semiconductor
D4 à D11, D15 = LED amarillo, baja corriente, SMD 0805, p.ej. Kingbright KP-2012SYCK
D1, D2, D3, D12, D14, D16 = LED naranja, baja corriente, SMD 0805, p.ej. Kingbright KPHCM-2012SECK
D13 = LED rojo, baja corriente, SMD 0805, p.ej. Kingbright KPHCM-2012SURCK

Varios:
K1 = Conector macho (divisible), 1 fila de 6 contactos con paso de 2,54 mm
K2 = Conector macho (divisible), 1 fila de 4 contactos con paso de 2,54 mm

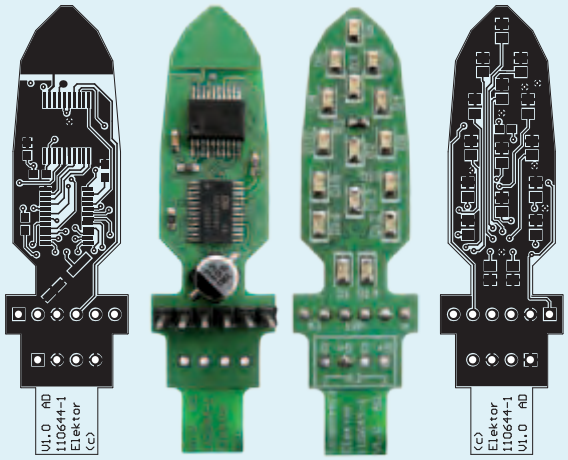


Figura 3. La placa de doble cara sólo lleva componentes de montaje superficial (SMD) pero, a pesar de ello, su montaje se puede hacer a mano.

este terminal está regulada a 1,2 V. Con R4 = 27 kΩ tenemos una corriente de 44 µA en la resistencia, con lo que tendremos 2,2 mA en los LEDs. Este valor proporciona una buena visibilidad de las animaciones manteniendo una buena visibilidad global. Si se utilizan otros LEDs, puede que sea necesario ajustar el valor de R4.
El CAT4016 se controla por una conexión serie. Se utilizan cuatro señales desde el seno de la CPU hacia el CAT4016:

- CLK: señal de reloj.
- SIN: señal de datos (*serial in*).
- LATCH: señal de memorización.
- BLANK: señal de apagado de los 16 LEDs simultáneamente.

La ficha técnica del componente muestra la secuencia de las señales para el control individual de los 16 LEDs. Los datos recibidos, sincronizados por el reloj, son almacenados en un registro de desplazamiento a cuya salida tenemos acceso (señal SOUT, *serial output*), que es de la que nos hemos servido aquí, pero que permite conectar varios componentes en serie sin consumir más entradas/salidas en el microcontrolador.
El CAT4016 efectúa un control de "todo o nada". Pero para hacer las animaciones algo más fluidas, necesitamos poder intervenir en la luminosidad de los LEDs y de manera individual. La solución es un programa.
Nuestro microcontrolador trabaja con una frecuencia de 16 MHz y, como son necesarios cuatro ciclos de reloj para ejecutar una instrucción, tenemos una frecuencia real de 4 MHz. El Timer 1 ha sido ajustado para producir una interrupción cada milisegundo. Esta fuente de cronometrado principal se denomina a menudo "*sistema tick*". Con cada *tick* refrescamos el contenido del registro de desplazamiento de IC2. En nuestra maqueta, el hecho de transferir los comandos para los 16 LEDs nos lleva 160 µs, es decir, el 16 % del tiempo del procesador
Con este ritmo, las sucesiones de estados de encendido y apagado de los LEDs no son perceptibles por el ojo humano. Así pues, aún nos queda bastante y tendremos que reducir las frecuencias de refresco, por ejemplo, añadiendo más tiempo de CPU a las otras rutinas del programa. De esta forma podemos crear una modulación de ancho de pulso (MAP) por programa para cada LED. En nuestro caso, nos hemos contentado con una MAP de cinco estados (0 %, 25 %, 50 %, 75 % y 100 %). Esto puede parecer poco, pero el programa también debe definir los motivos que van a describir los LEDs y los parpadeos parciales que intentan reproducir la vacilación de una llama real. Ahora bien, estamos limitados en espacio por nuestro microcontrolador. A menos que le dotemos de una memoria externa y de un método de programación de este espacio de almacenamiento, tenemos que disminuir el número de estados para poder definir suficientemente los motivos, con el fin de que nuestra animación sea atractiva.

La placa

Hemos diseñado una placa de 17 x 60 mm (ver **Figura 3**), en la que la zona ocupada por los LEDs mide 17 x 39 mm. La parte inferior de la placa está reservada para el conector de programación del microcontrolador y para los puntos de las entradas de alimentación.
Sobre la superficie superior de la placa sólo aparecen los LEDs con encapsulado SMD 0805, es decir, un tamaño de 2 x 1,25 mm. Sobre esta cara también encontramos el sensor de temperatura y la resistencia NTC en encapsulado 0603. ¡Es necesaria para poder soplar sobre ella!
Sobre la cara inferior, resalta rápidamente el montículo de nuestra placa que representa la tecla táctil. También conviene referenciar, debajo del micro controlador, el punto de prueba TP1 que ya se ha mencionado anteriormente.
Para alojar las otras funciones en la otra cara ha sido necesario recurrir a encapsulados de paso más pequeño. Nuestro microcontrolador se presenta en encapsulado SSOP20 con un paso de 0,65 mm y el controlador de los LEDs, que ha sido elegido con un encapsulado TSSOP24, también tiene un paso de 0,65 mm. Estos pasos más estrechos no deben ser vistos como un obstáculo, sino más bien como un esfuerzo para acercar las técnicas industriales a las posibilidades de un técnico electrónico particular. Así, las herramientas básicas serían una lupa con iluminación, un soldador con punta fina y ajustable en temperatura, estaño de diámetro de 0,5 mm y, sobre todo, ¡"flux" en jeringa! En cualquier caso, mejor no abusar del café antes de comenzar con el cableado...
La técnica es siempre la misma: hay que colocar una pequeña cantidad de estaño en la superficie de uno de los extremos, seguidamente, colocar el componente sobre sus terminales de soldadura y soldar uno de sus

A continuación se muestra el código correspondiente a la inicialización del temporizador 1 (Timer 1) para la base de tiempos de 1 KHz:

```
T1CON = 0b01100101 ; // configuración de Timer 1 (Tick Interrupt Source)
// 01       : la fuente de Timer1 es Fosc = 16 MHz
//   10     : Prescaler = 4 -> Ftimer = 4 MHz
//     0    : circuito oscilador Timer1 deshabilitado
//      1   : no sincronizar la entrada de reloj externo
//       0  : no implementado
//        1: TMR1ON=1 -> Timer1 habilitado

T1GCON = 0b00000000 ; // puerto de control de Timer 1
//  0       : TMR1GE=0 -> Timer 1 cuenta sin tener en cuenta las funciones Gate

TMR1H = 0xF0 ; // 0xF05F es el complemento de 0x0FA0=4000
TMR1L = 0x5F ; // valor inicial del contador para 1ms de pedido de interrupción (1000 Hz)
PIE1 = 0b00000001 ; // fuentes de interrupción
//        1: TMR1IE=1 -> fuente Timer1 IT habilitada
```

Seguidamente, el extracto del código para la inicialización de la función de gestión de la tecla capacite:

```
OPTION_REG = 0b01101000;
// 0        : WPUEN#=0 -> Weak pull-ups están habitadas por WPUx individuales
//  1       : Interrupción en flanco ascendente de INT (no usado)
//   1      : TMR0CS=1 -> Timer 0 cuenta en T0CKI
//    0     : TMR0SE=0 : Timer 0 cuenta en el flanco de subida de T0CKI
//     1    : PSA=1 : prescaler no está asignado a Timer0
//      000: valor de Prescaler, 1:2

// Configuración de la detección capacitiva
CPSCON0 = 0b10000101 ;
// 1        : módulo capacitivo habilitado
//  000     : no implementado
//     01   : oscilador en rango bajo
//       0  : estado de la dirección de corriente (sólo lectura)
//        1: TOXCS=1 -> entrada de Timer0 es la salida del Oscilador Capative

CPSCON1 = 0b00000000 ; // la fuente es CPS0/RA0
```

terminales estañado. Una vez que estimamos que ha sido bien colocado, se añade el "flux" y se sueldan el resto de los terminales. Es probable que se produzcan puentes entre los terminales, pero podemos desembarazarnos de ellos fácilmente con la ayuda de "trencilla" para desoldar.

El "rutado" de la placa se ha realizado en doble cara usando taladros pasantes metalizados. Las alimentaciones han sido "rutadas" en forma de planos: el plano de + 5 V en la cara superior, ya que esta tensión es el punto común de nuestros LEDs; y el plano de 0 V en la cara inferior.

El conector de programación "in-situ", K1, es, de hecho, una fila de seis taladros metalizados. La distribución de terminales sigue el orden de las señales dadas por el interfaz de programación ICD2 de Microchip (ver **Tabla 1**).

Hay una serie de puntos para el conector de alimentación K2: dos para los + 5 V (terminales 1 y 3) y dos para los 0 V (terminales 2 y 4). En este caso también el terminal 1 viene referenciado por una marca cuadrada. El doblar los terminales de alimentación permite, de manera práctica, conectar varias placas entre ellas para multiplicar el efecto gráfico.

Tabla 1. Distribución de terminales del conector ICSP. El terminal 1 viene referenciado por una marca cuadrada.

Contacto	Señal
1	MCLR/Vpp
2	Alimentación +5 V
3	Alimentación 0 V
4	ISPDAT
5	ISPCLK
6	Inutilizada (señal PGM)

El programa

A título indicativo, el código actual escrito en C ocupa el 84,5 % de la memoria flash del microcontrolador, es decir, 3460 palabras de las 4096 disponibles. Buena parte de las tres cuartas partes de esta memoria está dedicada a la definición de los motivos y aceptaría una cierta optimización si fuese necesario intervenir sobre este punto.

El código fuente completo, así como el fichero hexadecimal, están disponibles gratuitamente en la página de Internet del montaje. También podemos encontrar ahí los ficheros de la placa de circuito impreso.

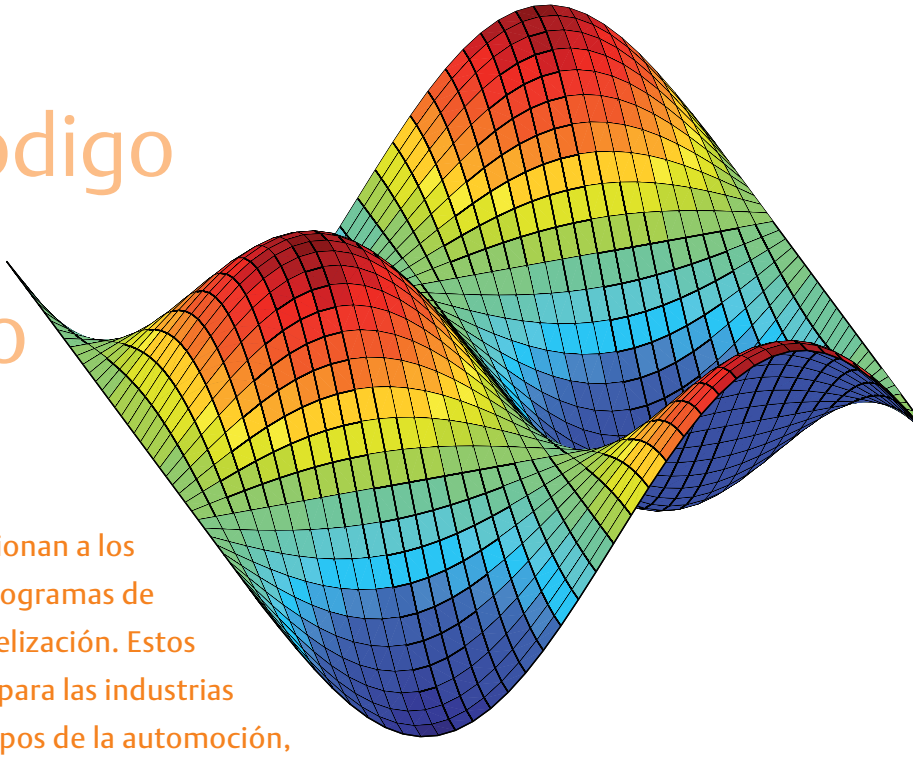
¡A vuestros soldadores y... Feliz Navidad!

(110644)

Enlaces Internet:

www.elektor.es/110644

Scilab, la elección de código abierto para el cálculo numérico



Vincent Couvert, Bruno Jofret y Julie Paul (Francia)

Los programas de cálculo numérico proporcionan a los ingenieros un conjunto de herramientas y programas de diseño y estudio para la simulación y la modelización. Estos programas se han convertido en ineludibles para las industrias que han recurrido a la simulación en los campos de la automoción, aeronáutica, energía, química, finanzas... Estos programas les permiten limitar, e incluso, ahorrar pruebas costosas y exigentes que habría que establecer en situaciones reales.

Para mantenerse en la carrera de una competitividad creciente y responder a la preocupación exacerbada y generalizada de reducción de costes, el mundo industrial se interesa cada vez más en los programas que compiten, muy seriamente, con los programas propietarios que están acostumbrados a usar. Pero hoy día, la elección de un programa de código abierto no sólo concierne únicamente al dominio del gran público en la navegación por Internet o en el tratamiento de textos (como la adopción de OpenOffice.org por numerosas administraciones públicas), los programas del mundo de la industria se han visto igualmente afectados.
Soportado por un consorcio de usuarios industriales y utilizado en el mundo entero, Scilab [1] constituye hoy día una alternativa creíble a Matlab [2]. Además del beneficio nada despreciable de su coste, el acceso y el control completo del código fuente constituyen, a menudo, un argumento decisivo en su adopción por los numerosos usuarios y los sectores estratégicos de defensa o de la aeronáutica. El mundo académico y el de la educación hace tiempo que ya han adoptado a Scilab que, por ejemplo, ha sido reconocido de interés pedagógico en junio de 2011 por el Ministerio de Educación nacional francés.

El programa Scilab en la práctica

Scilab es un programa libre distribuido bajo licencia CeCILL (compatible GPL). Disponible para los sistemas de explotación más corrientes (Windows, Mac y Linux), se puede descargar gratuitamente desde [1].
Scilab es un entorno completo, abierto y extensible, para el cálculo y la visualización. Basado en el cálculo matricial, el programa integra centenares de funciones matemáticas y un lenguaje de programación con grandes prestaciones. Ofrece la posibilidad de conexión con otros programas escritos en distintos lenguajes (C, C++, Java). La sintaxis de Scilab es comparable a la de Matlab en numerosos puntos, sin ser compatible al 100 %. Su espectro funcional es, igualmente, muy amplio y el usuario puede añadir en el mismo numerosos módulos externos de simulación, visualización gráfica, optimización, estadísticas, diseño y estudio de sistemas y de control, tratamiento de la señal, etc.
Como muchos de los programas libres, Scilab es interoperable. El usuario puede adaptarlo a sus necesidades e incrementar sus funcionalidades nativas. Por ejemplo, la pasarela entre LabVIEW [3] de National Instruments y Scilab permite la puesta en funcionamiento de un tratamiento de datos completo y potente. Los usuarios escriben sus “scripts” Scilab directamente en LabVIEW y llaman a Scilab para analizar y visualizar sus datos.

Un entorno integrado

Scilab ofrece un entorno de trabajo integrado y ergonómico que facilita su aprendizaje y su uso. Su consola permite un uso sencillo mostrando líneas de comando y de resultados. Su editor de texto y sus funciones avanzadas permiten almacenar el conjunto de sus programas y volver a encontrar su entorno de trabajo en una nueva ejecución. Del mismo modo, proporciona una interacción total con la consola, permitiendo al usuario, por ejemplo, ejecutar una parte o la totalidad de un fichero que se está editando.

Funciones de visualización avanzadas

Las funciones gráficas 2D y 3D han sido incluidas para visualizar, anotar y exportar datos. Ofrece numerosas formas de crear y de

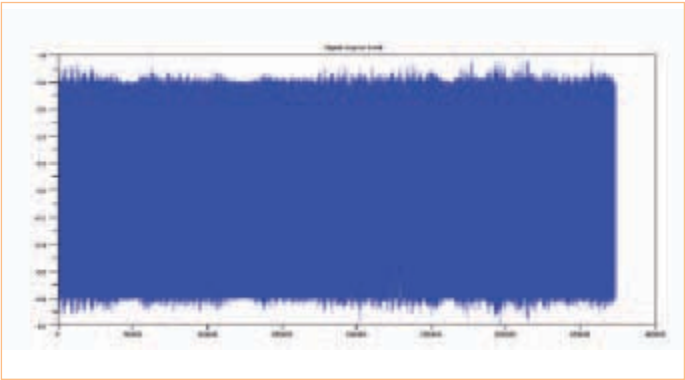


Figura 1. La señal de entrada.

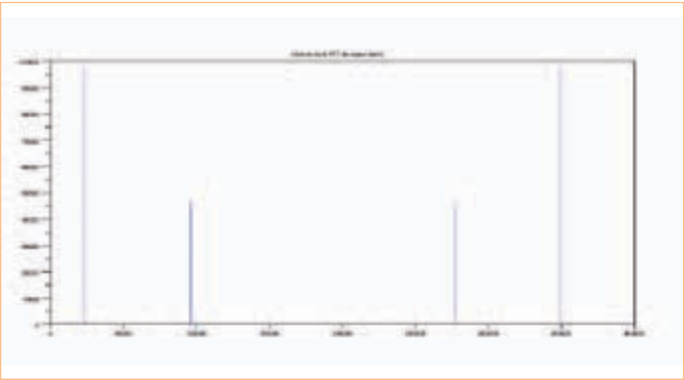


Figura 2. La transformada de Fourier de la señal de entrada muestra que está perturbada por dos frecuencias distintas.

personalizar diferentes tipos de trazas y de diagramas. Los resultados calculados por el usuario en la consola de Scilab pueden, de esta forma, ser visualizados de manera interactiva.
Para profundizar sobre las funciones del programa, se han propuesto dos ejemplos de su uso en el dominio del tratamiento de la señal.

Ejemplo 1: Filtrado digital con Scilab

En este ejemplo, Scilab es utilizado para filtrar una señal proveniente de un fichero de audio de tipo WAV. Todas las herramientas necesarias para realizar este trabajo están incluidas en Scilab: lectura/escritura de ficheros, cálculo de la transformada rápida de Fourier, cálculo y aplicación de filtros digitales.
El fichero de sonido, que hemos colocado en el directorio actual de Scilab (que se puede consultar y modificar llamando a la función `cd` de Scilab) y que se llama «SonBruite.wav» [4], es cargado con la función `loadwave`, la cual devuelve los datos sonoros bajo la forma de un vector, así como las informaciones sobre el fichero (frecuencia de muestreo, etc.). En el editor de texto, introducimos las líneas siguientes antes de ejecutarlas (menú Ejecutar/hasta cursor con eco):

```
stacksize(«max»); // Aumento de la memoria asignada a Scilab
[signalBruite, infosSignal] = loadwave(«SonBruite.wav»);
frequenceEchant = infosSignal(3)
nbEchantillons = infosSignal($)
```

Lo que se muestra en la consola es:

```
-->frequenceEchant = infosSignal(3)
 frequenceEchant  =

    22050.
```

```
-->nbEchantillons = infosSignal($)
 nbEchantillons  =

    373380.
```

Los resultados obtenidos indican que la señal ha sido muestreada a una frecuencia de 22.050 Hz y que nuestra señal contiene 373.380 muestras (valores). La función *plot* permite visualizar esta señal (ver **Figura 1**):

```
plot(signalBruite)
xtitle(«Signal original bruité»);  // Título del gráfico
```

Para conocer las frecuencias que perturban a la señal útil, calculamos la Transformada de Fourier de la señal con la función `fft`, después, calculamos su módulo usando la función `abs`:

```
moduleFftSignalBruite = abs(fft(signalBruite));
scf(); // Apertura de una nueva ventana gráfica
plot(moduleFftSignalBruite);
xtitle(«Module de la FFT du signal bruité»)
```

Con la función `plot`, obtenemos el gráfico de la **Figura 2**. Dicho gráfico indica que esta señal simétrica está perturbada por dos frecuencias distintas. Serán necesarios dos filtrados sucesivos para volver a tener la señal útil. La determinación precisa de las frecuencias de ruido queda simplificada en Scilab con sus numerosas funcionalidades en cálculo matricial: operaciones matemáticas, identificación del valor máximo y de su posición en los datos...
La ejecución del siguiente código desde el editor de textos permite definir la frecuencia correspondiente al pico:

```
// La transformada de Fourier es simétrica
// De aquí en adelante, sólo se guarda la primera mitad de los puntos de frecuencias = frequenceEchant*(0:(nbEchantillons/2))/nbEchantillons;
```

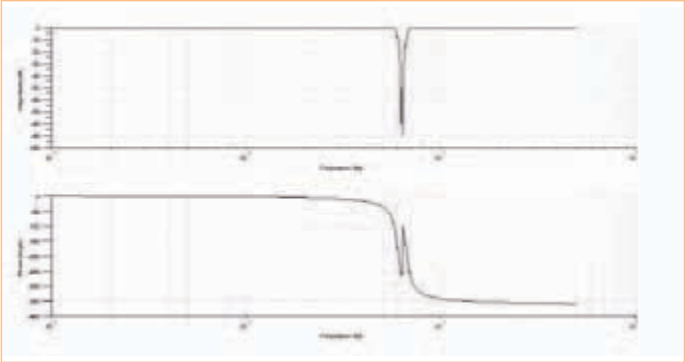


Figura 3. Diagrama de Bode del filtro IIR que permitirá suprimir la señal de 1397 Hz.

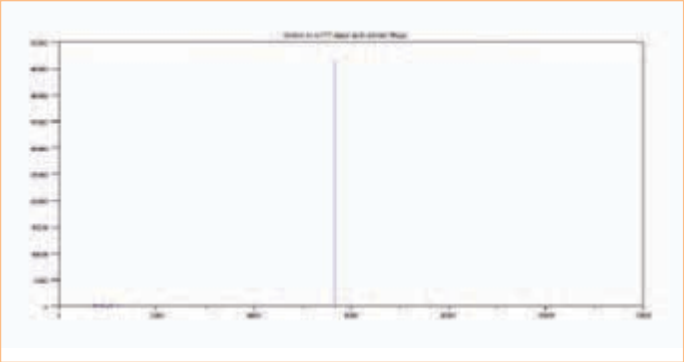


Figura 4. Después del filtrado, la señal de entrada sólo está perturbada por una señal parásita.

```
// Búsqueda del valor de pico a filtrar
[valeurPic, indicePic] = max(moduleFftSignalBruite(1:
size(frequences, «*»)));
indicePic
frequences(indicePic)
```

Verificamos los resultados en la consola:

```
-->indicePic
 indicePic  =

    23657.

-->frequences(indicePic)
 ans  =

    1397.0079
```

El primer pico representa el ruido de una frecuencia de 1.397,0079 Hz. Un filtro de banda de corte (Butterworth de orden 3), centrado sobre esta frecuencia y de un ancho de banda de 200 Hz, permite suprimir este ruido. La función de transferencia de este filtro se obtiene con la función `iir`:

```
hz = iir(3, «sb», «butt», [frequences(indicePic)-100
frequences(indicePic)+100]/frequenceEchant, [0 0]);
```

Para visualizar simplemente el diagrama de Bode del filtro, utilizamos la función *bode* (ver **Figura 3**). Sólo nos queda aplicar este filtro a la señal perturbada con una sencilla línea de código:

```
// Filtrado de la señal original perturbada
signalFiltre = filter(hz.num, hz.den, signalBruite);
```

La señal obtenida sigue estando perturbada por la segunda frecuencia, tal y como lo muestra la transformada de Fourier de la **Figura 4** (calculada utilizando el mismo método que anteriormente). Esta gráfica permite validar que ya se ha suprimido correctamente el primer pico. Ahora es necesario filtrar el segundo pico con la ayuda de otro filtro de banda de corte (por ejemplo, un filtro Chebyshev tipo I de orden 3), centrado sobre la frecuencia correspondiente a dicho pico y utilizando la misma técnica que anteriormente:

```
// Búsqueda del valor de pico a filtrar
[valeurPic, indicePic] = max(moduleFftSignalFiltre(1:
size(frequences, «*»)));
// Cálculo del filtro Chebyshev correspondiente
hzFiltre2 = iir(3, "sb", "cheb1",
[frequences(indicePic)-100 frequences(indicePic)+100]/
frequenceEchant, [0.01 0]);
// Filtrado de la señal una vez
signalFiltre = filter(hzFiltre2.num, hzFiltre2.den,
signalFiltre);
```

Visualizamos el resultado (ver **Figura 5**):

```
scf();
plot(signalFiltre)
xtitle(«Signal après second filtrage»)
```

A continuación, la señal obtenida es filtrada completamente y un último cálculo de la transformada de Fourier de la señal muestra que las dos frecuencias que perturbaban nuestra señal original han desaparecido completamente (ver **Figura 6**):

```
moduleFftSignalFiltre = abs(fft(signalFiltre));
scf();
plot(frequences, moduleFftSignalFiltre(1:size(frequen
ces, «*»)));
xtitle(«Module FFT signal après second fitrage»)
```

Almacenamos nuestro resultado en un fichero WAV:

```
savewave(«SonApresFiltrage.wav», signalFiltre)
```

Ejemplo 2: Detección de contornos

En el dominio del tratamiento de la imagen o, más ampliamente, de la visión por ordenador, la detección de los contornos es una de las primeras etapas necesarias para la utilización de algoritmos más complejos de detección de objetos, detección de caras, (*face tracking*)... Basta con representar la imagen bajo la forma de una matriz de valores (generalmente los niveles de gris) y aplicar sobre esta matriz los tratamientos deseados. Scilab juega aquí un doble papel ya que permite, a la vez, visualizar la imagen así como sus evoluciones a lo largo de sus transformaciones y tratamientos, y manipular los datos utilizando las capacidades de cálculo matricial que ofrece.

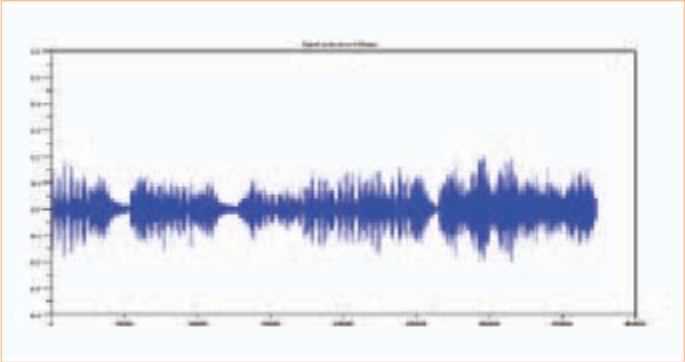


Figura 5. El filtrado ha permitido suprimir las dos señales parásitas.

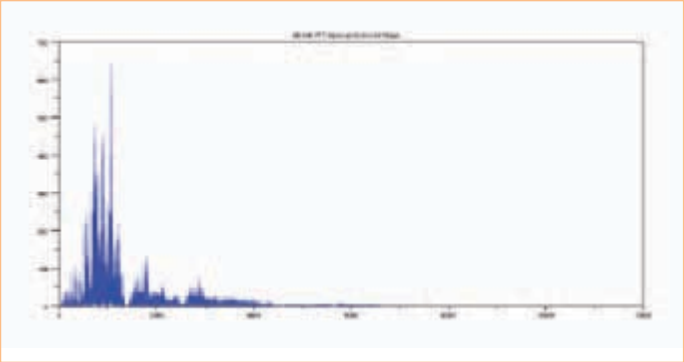


Figura 6. Transformada de Fourier de la señal de entrada filtrada.

Este ejemplo aplica un método del primer orden (gradiente), puesto en funcionamiento con la ayuda de filtros de Prewitt, Sobel y Scharr. La carga de una imagen con el formato PGM (*portable graymap*, imagen en nivel de gris), que se nombra como «Scilab.pgm» [4], se realiza por la función `readimage` [4]. Esta función abre el fichero, lee el tamaño de la imagen, después el nivel de gris asociado a cada píxel (comprendido entre 0 y 255) y devuelve la imagen bajo la forma de una matriz.

```
stacksize(«max»); // Augmentation de la mémoire allouée
à Scilab
gray_m = readimage(«Scilab.pgm»);
```

A lo largo de este ejemplo, y para mayor sencillez, utilizaremos la función `showImage` para visualizar en nivel de gris todas las futuras transformaciones. Esta función se apoya en la función nativa de Scilab, `Matplot` [5], que permite mostrar una matriz. Utilizando la tabla de niveles de grises correspondiente, podemos visualizar la imagen:

```
function []=showImage(imageMatrix)
   f = figure();
   f.color_map = graycolormap(255);
   f.background = -2;
   Matplot(imageMatrix);
  endfunction

 showImage(gray_m);
```

La imagen cargada en Scilab se representa en una ventana gráfica (ver **Figura 7**).
La detección de los contornos sobre una imagen en niveles de grises se efectúa gracias a unos cálculos sobre una matriz 2D. Es posible trabajar con matrices de N dimensiones para representar imágenes a color (RGB, RGBA, CMYK, HSV...). Los tratamientos a aplicar se harían sobre cada uno de los canales, incluso combinados entre ellos para obtener otros resultados.
La etapa siguiente para la detección de contornos consiste en aplicar un «alisado» de la imagen para disminuir el ruido de la imagen de origen y obtener así mejores resultados en la detección final. Scilab permite definir la función de alisado utilizando un producto de convolución y una matriz gausiana 2D. Dicha función toma como parámetro la matriz que representa la imagen y devuelve otra matriz que representa la imagen «alisada».

Se añaden dos rangos de píxel de forma artificial, al borde de la imagen, para implementar el producto de convolución con la ayuda de la función `sum` y de un producto de matrices. Estas funciones están optimizadas en Scilab y son más potentes que si se efectuase cada una de las operaciones, elemento por elemento.

```
function N=blurr(P)
    N = zeros(P);
    P2 = [zeros(1,size(P,»c»)); zeros(1,size(P,»c»));
P; zeros(1,size(P,»c»)); zeros(1,size(P,»c»))]
    P2 = [zeros(size(P2, «r»), 1), zeros(size(P2, «r»),
1), P2, zeros(size(P2, «r»), 1), zeros(size(P2, «r»),
1)]
    K = 1/159 * [2  4  5  4 2
    4  9 12  9 4
    5 12 15 12 5
    4  9 12  9 4
    2  4  5  4 2];

    for x=3:(size(P2, «r») - 2)
        for y=3:(size(P2, «c») - 2)
            r = 0;
           N(x-2,y-2) = sum(K .* P2(x-2:x+2, y-2:y+2));
        end
    end
endfunction
```

Una implementación más ingeniosa del producto de convolución, tal como existe en la literatura, permitiría obtener resultados numéricamente equivalentes, pero con tiempos de cálculo bastante más largos.

```
function N=dummy_blurr(P)
    N = zeros(P);
    K = 1/159 * [2  4  5  4 2
    4  9 12  9 4
    5 12 15 12 5
    4  9 12  9 4
    2  4  5  4 2];
    for x=1:size(P, «r»)
        for y=1:size(P, «c»)
            r = 0;
            for i = -2:2
                for j = -2:2
```

Figura 7. Imagen original en nivel de gris.

Figura 8. Imagen original alisada para facilitar la detección del entorno.

```
                    if (x + i > 0 & x + i <= size(P,
«r») & y + j > 0 & y + j <= size(P, «c»))
                        r = r + K(i+3, j+3) * P(x+i,
y+j)
                    end
                end
            end
            N(x,y) = r;
        end
    end
endfunction
```

La imagen «alisada» así obtenida (ver **Figura 8**), que aparece borrosa a vista normal, va a ser utilizada para los cálculos del gradiente que pondrán en evidencia los contornos de dicha imagen.
Para estos cálculos del gradiente, hemos utilizado los filtros de Sobel (ver **Figura 9**) y de Prewitt (ver **Figura 10**). Estos dos filtros son realizados por un producto de convolución que sirve para calcular un gradiente.

```
function N=convol2d(K, P)
    N = zeros(P);
    P2 = [zeros(1,size(P,»c»)); P; zeros(1,size(P,»c»))]
     P2 = [zeros(size(P2, «r»), 1), P2, zeros(size(P2,
«r»), 1)]
    for x=2:(size(P2, «r») - 1)
        for y=2:(size(P2, «c») - 1)
            r = 0;
           N(x-1,y-1) = sum(K .* P2(x-1:x+1, y-1:y+1));
        end
    end
endfunction
```

Para Sobel :

```
GX = convol2d([-1 0 1 ; -2 0 2 ; -1 0 1], gray_m);
GY = convol2d([-1 -2 -1 ; 0 0 0 ; 1 2 1], gray_m);
contourSobel = sqrt(GX.^2+GY.^2);
showImage(contourSobel);
```

Scilab no es el único programa de cálculo numérico gratuito y de código abierto

He aquí otros tres ejemplos que merecen ser mencionados:

– **Octave** es un lenguaje de programación interpretado y estructurado como el lenguaje C, que acepta numerosas instrucciones de la librería estándar de C. Puede ser ampliado para aceptar las funciones y las llamadas al sistema Unix y también pueden ser utilizadas las funciones escritas en C++. Para la mayoría de los comandos, la sintaxis es la misma que la de MATLAB y, una programación cuidadosa, permite hacer funcionar los "scripts" tanto sobre Octave como sobre MATLAB. (fuente: Wikipédia)

http://www.gnu.org/software/octave/

– **FreeMat** es un entorno de cálculo informatizado y un lenguaje de programación bajo la forma de un programa libre, relativamente compatible a nivel de sus fuentes con Matlab y Octave. Acepta fácilmente código externo escrito en C, C++ y Fortran. También ofrece la posibilidad de desarrollo de algoritmos distribuidos paralelos y también posee algunas capacidades de renderización de volúmenes y de visualización 3D. La versión actual 4.0 data de octubre de 2009. (fuente: Wikipédia)

http://freemat.sourceforge.net

– **JMathLib** se presenta como clon en java de SciLab, Octave, Freemat y Matlab. Como Freemat, este proyecto parece menos activo que Scilab y Octave. La versión actual 0.9.4 data de febrero de 2009.

http://www.jmathlib.de/

Para Prewitt :

```
GX = convol2d([-1 0 1 ; -1 0 1 ; -1 0 1], gray_m);
GY = convol2d([-1 -1 -1 ; 0 0 0 ; 1 1 1], gray_m);
contourPrewitt = sqrt(GX.^2+GY.^2);
showImage(contourPrewitt);
```

Otras fórmulas dan resultados más satisfactorios utilizando otro modelo de convolución: Costella, Robert Cross y Scharr. Su implementación en Scilab es fácil gracias a un lenguaje simple y bastante próximo al de las matemáticas (sin declaración de variables, sin tipo, sin asignación de memoria, etc.). Por ejemplo, el filtro de Scharr (ver **Figura 11**) viene implementado por las líneas de código siguientes:

```
GX = convol2d([3 0 -3 ; 10 0 -10 ; 3 0 -3], gray_m);
GY = convol2d([3 10 3 ; 0 0 0 ; -3 -10 -3], gray_m);
contourScharr = sqrt(GX.^2+GY.^2);
showImage(contourScharr);
```

Esta matriz puede servir de base a otros algoritmos de detección de formas. Esto no es más que un primer paso en el tratamiento de la imagen. Se podría ir más lejos realizando filtrados suplementarios sobre valores umbrales. Los contornos más francos, correspondientes a las zonas más claras, serían conservados y el resto eliminados. La matriz así obtenida contendría únicamente “0” y “1” para cada píxel reconocido como formando parte de un contorno.

Los ejemplos de utilización de Scilab son infinitos y cubren los dominios de aplicaciones variadas. Os proponemos encontrarnos en un próximo número de Elektor, con un artículo sobre Xcos, la herramienta de Scilab para la modelización y la simulación de sistemas dinámicos híbridos, equivalente al Simulink de Matlab.

(110491)

Figura 9. Contornos puestos en evidencia por el filtro de Sobel.

Figura 10. Contornos encontrados por el filtro de Prewit.

Figura 11. Contornos según el filtro de Scharr.

Enlaces en internet

[1] Scilab: www.scilab.org

[2] Matlab: www.mathworks.com/matlab

[3] LabVIEW: www.ni.com/labview

[4] Ficheros de los ejemplos presentados: www.elektor.es/110491

[5] Ayuda en la función Matplot: http://help.scilab.org/docs/current/fr_FR/Matplot.html

La PCB Prototyper en la práctica

Ideal para prototipos y pequeñas series

La PCB Prototyper que Elektor introdujo hace un año ha sido ya adquirido por muchos laboratorios y empresas. Esta máquina permite fresar una placa de una cara o doble cara de forma sencilla sin necesidad de tener un amplio conocimiento en fresadoras. Fuimos a verlo a casa de un usuario de PCB Prototyper.

Harry Baggen (Redacción Holanda)

Poco después de que Elektor presentara la PCB Prototyper en diciembre de 2010, recibimos rápidamente los primeros pedidos. Al mismo tiempo, el fabricante Colinbus trabajaba en la primera serie de producción y poco después se empezaban a suministrar las primeras unidades. Las reacciones de los usuarios resultaron ser muy positivas. No se esperaba una máquina tan precisa y de uso tan amigable por un precio tan módico (las fresadoras son normalmente mucho más caras que los 3500 euros – IVA excluido – que cuesta la PCB Prototyper).

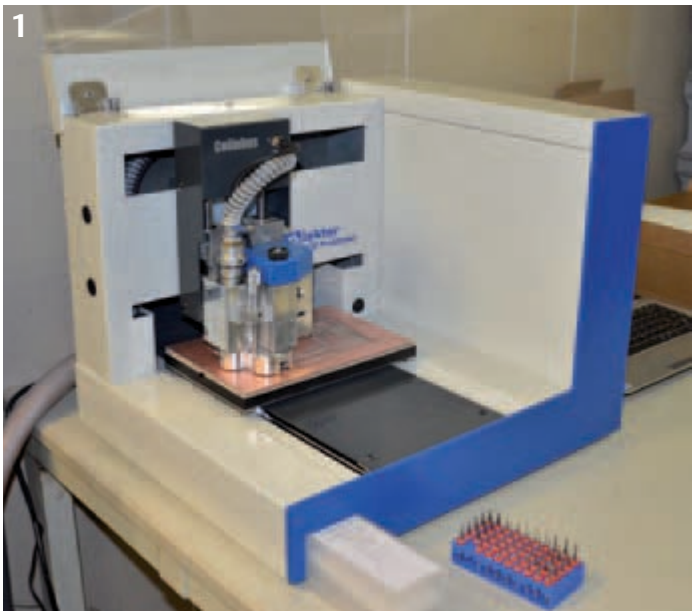
Aquí en la redacción sentimos mucha curiosidad por cómo se utiliza este tipo de máquinas en la práctica, y a raíz de esto planificamos una visita a la empresa Avasto en Oudewater, que ya llevaba trabajando algún tiempo con una PCB Prototyper.

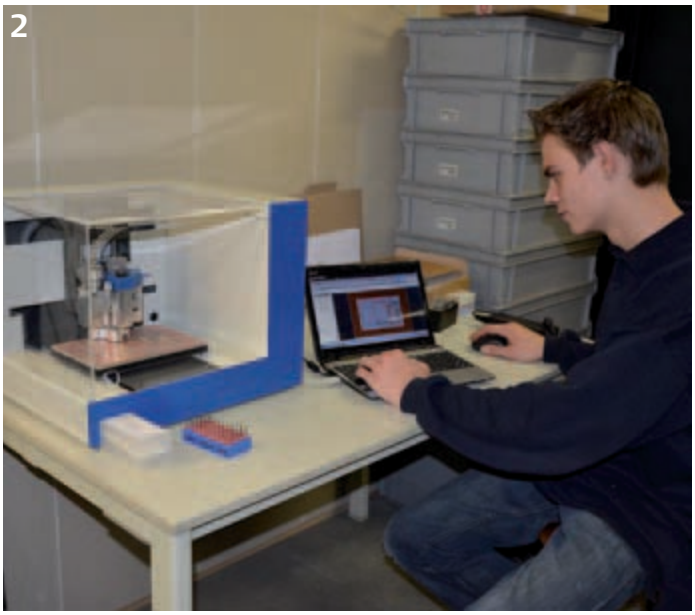
Avasto es una polifacética empresa de instalación dedicada a trabajos de construcción, automatización de procesos de producción y diseño y mantenimiento de saunas, jacuzzi, etc. También se utiliza electrónica de control en diferentes proyectos, muchas veces a base de PLC. Sin embargo, la empresa está añadiendo cada vez más desarrollos propios de electrónica en sus productos. El producto más nuevo es una protección de tobogán para ser utilizado en piscinas. Este sistema indica al principio del tobogán cuando puede montar el siguiente nadador. El nadador puede iniciar el cronometraje mediante un pulsador. Después de abandonar el tobogán en la parte de abajo de la piscina, se pulsa sobre otro pulsador de finalización y aparece el tiempo transcurrido en un marcador. De esta manera no sólo se evitan taponamientos en el tobogán, sino que también se introducen elementos de competición de modo que el tobogán resulta más atractivo.

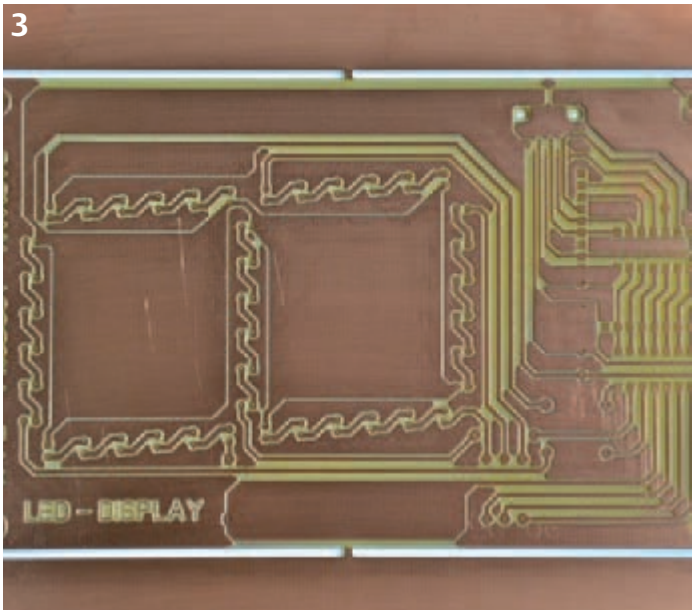
Avasto diseñó por completo este sistema en su propia casa y ya ha instalado varios ejemplares. El sistema está incluso verificado por un instituto de marca de calidad.

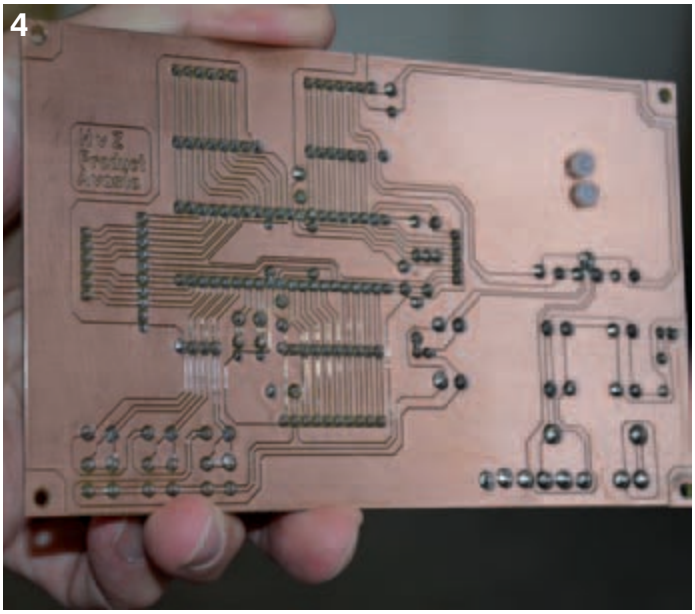
El copropietario, Swen van Vrouwerff, es un técnico de pura cepa que conoce muy bien los proyectos en los que está trabajando la empresa, tanto a nivel mecánico como a nivel electrotécnico y electrónico. En la actualidad su empresa desarrolla cada vez más circuitos electrónicos por cuenta propia. Cuando Elektor presentó la PCB Prototyper, le pareció la máquina ideal para su empresa. De esta forma puedes producir placas para una pequeña cantidad de productos. Bien es verdad que puedes sopesar pedir pequeñas cantidades a un fabricante de PCB, pero esto cuesta más tiempo y relativamente más dinero. Swen está convencido de que se amortiza rápidamente la inversión.

En este momento la PCB Prototyper se utiliza principalmente para la producción de placas en la protección de toboganes previamente descrita. En una caja dura de la parte del display se encuentra toda la electrónica del sistema, repartida sobre 9 pla-


1


2





cas. La PCB Prototyper (1) produce todas las placas. Lo controla un netbook colocado al lado de la máquina (2). Digno de mención es el hecho de que dos jóvenes empleados con una formación profesional media controlen la PCB Prototyper y construyan la máquina de protección del tobogán al completo. Nos contaron que la PCB Prototyper es fácil de manejar. Después de experimentar durante un día pudieron trabajar ya con ella y fresar placas de excelente calidad. La producción de una placa de más o menos el formato de un eurocard tardó más o menos media hora. Como la máquina para automáticamente para cambiar de taladro o fresa no hace falta estar todo el tiempo pendiente y se pueden realizar otras tareas al mismo tiempo. En (3) ves el resultado final, en este caso una placa para un segmento del display.

A continuación se montan las placas y se las provee de una gruesa capa de plástico en el lado del cobre, ya que el cobre debe ir bien protegido contra las influencias del cloro, que en una piscina se encuentra por todas partes (4). La foto (5) muestra otra vez la calidad de las pistas fresadas en la placa. Finalmente la foto (6) muestra la caja de protección del tobogán con las placas montadas.

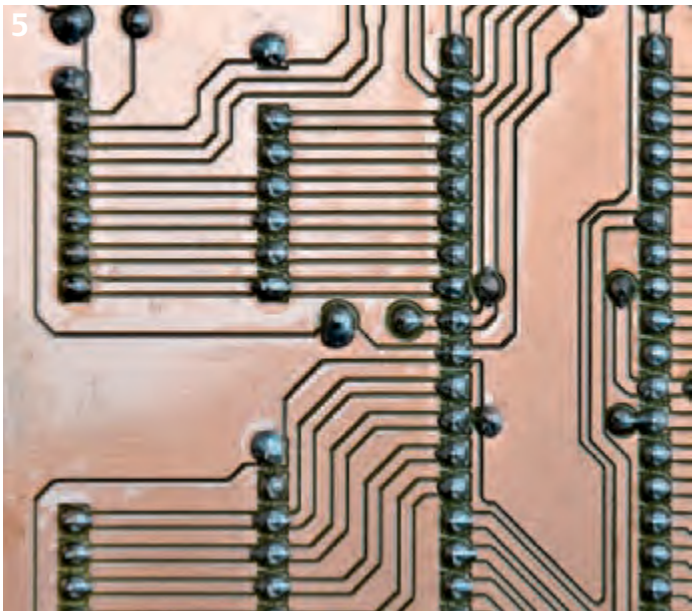
Esta es sólo una de las muchas posibilidades de aplicación de la PCB Prototyper. En este momento se está produciendo principalmente placas de una única cara para componentes de taladro pasante, pero si en el futuro Avasto quiere cambiar a SMD, ¡entonces la PCB Prototyper podrá fresar dichas placas sin ningún problema! Además, próximamente aparecerán varias opciones de ampliación para la PCB Prototyper, ¡con las que la máquina será aún más polifacética!

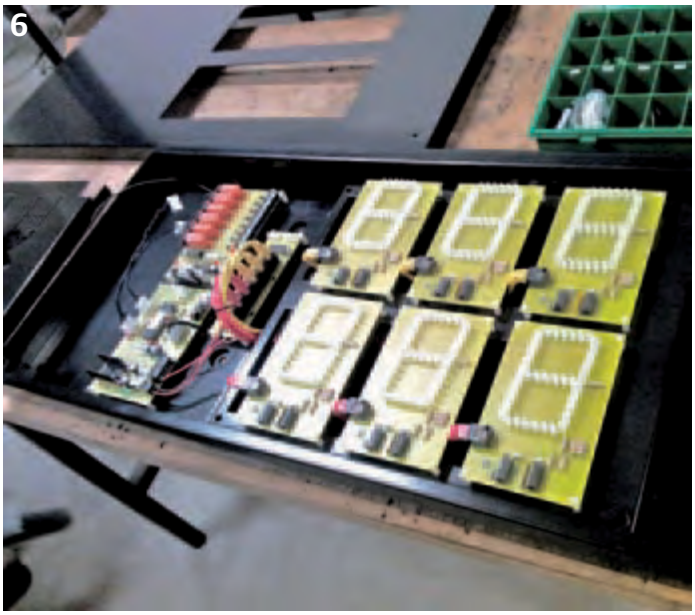
(110694)

Enlaces Web

www.elektor.es/proyectos/pcb-prototyper-(100619).1601209.lynkx

www.avasto.nl





Medidor de Gradiente de Temperatura

Detecta e informa de los cambios más pequeños

Dr Dietmar Schröder (Alemania)

A veces no es tan interesante conocer los valores absolutos de temperatura como los pequeños cambios que se producen. En el circuito que aquí presentamos no solo se detectan, sino que también se hacen visibles y audibles. Con solo unos pocos componentes activos, indica la temperatura medida con una resolución de una diezmilésima de grado.

La medición de la temperatura es una tarea bastante simple, pero el hacerlo con tanta precisión exige unos mejores conocimientos, en particular cuando se desean obtener lecturas a una resolución mejor de una décima de grado. El circuito que se presenta aquí mide la temperatura a una resolución de una diezmilésima de grado, usando sólo cuatro componentes activos y una pantalla de visualización opcional. El objetivo del circuito es ser capaz de detectar hasta el cambio más pequeño de temperatura: el énfasis se ha puesto mucho más en sensibilidad extrema y en la resolución que en la precisión absoluta.

Prestaciones y posibilidades

Sin ninguna calibración aburrida, el circuito mide la temperatura con una precisión absoluta básica de, aproximadamente, dos grados. También muestra el gradiente de temperatura o la velocidad de cambio de la temperatura con el tiempo. La salida se hace a través de una pantalla digital y una aguja analógica, a la vez que se representa con el cambio de tono de una salida acústica. Las lecturas también están disponibles en un puerto serie para su transferencia a un ordenador personal.

El instrumento se presta a una variedad de aplicaciones interesantes. Su sensibilidad es tan alta que su lectura muestra cambios claros de la temperatura cuando el sensor se sube o se baja, reflejando el hecho que, en un cuarto, el aire caliente se eleva hacia el techo. Si el sensor es colocado en la salida de aire del ventilador de un PC, es posible detectar cambios de la carga en su CPU o de su tarjeta gráfica, reflejados en la temperatura del aire arrojado. Otra opción es la de colocar el sensor sujeto a una tubería de casa para detectar cuando se está usando el agua.

La salida acústica permite la monitorización de la temperatura 'sin manos'. Por ejemplo, el sensor puede ser conectado a un componente delicado de un circuito para dar un aviso acústico inmediato cuando, de repente, comienza a disipar más potencia.

Es importante aislar bien el sensor para mantener la lectura estable. Por ejemplo, puede ser introducido en el fondo de una espuma de aislamiento, como el polietileno expandido. Si la constante de tiempo del gradiente de temperaturas se elije de forma apropiada, es incluso posible medir cambios lentos de la temperatura ambiente fiable a pesar del aislamiento.

El autor, al principio, usó el circuito para analizar el comportamiento de la válvula termostática en el sistema de refrigeración de un coche.

Construcción

Esencialmente, la unidad está formada por un termistor NTC, usado como sensor de temperatura, una tensión de referencia, un conversor A/D y un microcontrolador que controla una pantalla LCD.

Para reducir al mínimo la interferencia y el ruido, el termistor debe ser colocado lo más cerca posible del conversor A/D. Así, podemos trabajar con el sensor a una distancia relativamente grande del conjunto de la unidad. Para ello, el circuito está dividido en dos placas de circuito impreso.

El sensor se pega en el exterior de la placa del sensor, muy estrecha, que también aloja la tensión de referencia y el conversor A/D. El sensor se coloca, pues, a sólo unos 10 mm del conversor A/D. La placa del sensor puede ser alojada en un tubo radiador, formando todo el conjunto la sonda de temperaturas de la unidad (ver **Figura 1**).

Las lecturas son llevadas en formato digital a la placa del procesador (ver **Figura 2**), que contiene un microcontrolador ATmega88, sobre un cable de cuatro hilos, usando un protocolo serie. La placa dispone de espacio para conectar una pantalla LCD, un zumbador piezoeléctrico, un medidor analógico y un interfaz serie (o un cable interfaz USB-a-serie). El interfaz serie usa niveles de señal TTL. Tres potenciómetros permiten el ajuste de la sensibilidad básica, la constante de tiempo y el volumen del zumbador piezoeléctrico.

El circuito

En la **Figura 3** se muestra el esquema eléctrico de la placa del sensor. La alimenta-

Prestaciones

Rango de funcionamiento: 0 °C a 40 °C
Precisión absoluta: aproximadamente, 2 °C no calibrados
Resolución: en teoría, 4×10^{-5} °C, en la práctica, aproximadamente 3×10^{-4} °C; resolución mostrada: 10^{-4} °C
Rango: grados por 2 segundos a grados por 100 minutos
Tensión de alimentación: 6 V (cuatro pilas AA)

ción de 5 V es suministrada desde la placa del procesador, sobre el cable de interconexión de cuatro hilos y filtrada por la red RC formada por R100 y C100. Desde aquí, el diodo TL431, de la tensión de referencia, genera una referencia de bajo de 2,5 V, la cual es de nuevo filtrada por R102 y C103 y llevada al conversor analógico/digital MCP3551. El valor abosoluto de la tensión de referencia no es particularmente crítico, ya que el resultado de la conversión A/D sólo depende de la proporción entre la resistencia R103 y la del termistor. La resistencia R103 ha sido elegida de modo que la curva característica de los resultados de conversión resultantes sea más o menos lineal en el rango de temperatura de 0 °C a 40 °C: ver **Figura 4**. El condensador C104 en la entrada al conversor A/D funciona como un filtro de anti-aliasing (amortiguamiento).

El MCP3551 es un conversor delta-sigma con una resolución de 22 bits. En el rango de los 20 °C esto equivale a, aproximadamente, 24.000 pasos de cuantificación por grado de temperatura, dando una resolución (teórica, al menos) de 40 millonésimas de grado. Sin embargo, en la práctica, la componente del ruido limita las prestaciones a una resolución eficaz de, aproximadamente, 19 bits.

El MCP3551 lee la tensión en los extremos del termistor a unas 20.000 veces por segundo y filtra digitalmente los resultados. El chip no sólo contiene un filtro paso bajo sino también un filtro "notch" de conformación de onda, que suprime el zumbido de 50/60 Hz de la tensión de red. Está disponible una nueva lectura unas 14 veces por segundo en el interfaz serie, transfiriéndose dichas lecturas a la placa del procesador.

La **Figura 5** muestra el esquema eléctrico del circuito de la placa del procesador. El cable de cuatro vías de la placa del sensor está conectado al conector X202. Los conectores X201 y X205 han sido proporcionados para los potenciómetros P1 (10 k Ω) y P2 (también de 10 kΩ), que se utilizan para ajustar las constantes de tiempo del procesamiento digital hecho por el programa del microcontrolador. Un tercer potenciómetro permite el ajuste del volumen del zumbador piezoeléctrico (el cual no debería tener ningún circuito oscilador interno, y una capacidad de, aproximadamente, entre 8 nF y 20 nF). El conector para este potenciómetro, X206, tiene cinco terminales, permitiendo el uso de un potenciómetro de 10 kΩ con un interruptor incorporado que se puede usar para encender y apagar la unidad. Se utiliza un cuarto potenciómetro (P500) para configurar el contraste de la pantalla LCD contactada a X207. El autor utilizó una pantalla Wintek WD-C2704, que tiene cuatro filas de 27 caracteres y un controlador compatible HD44780, que puede hacerse funcionar en modo de cuatro bits o de ocho bits.

El ATmega88 puede funcionar con su oscilador interno de 1 MHz, evitando la necesidad de un cristal de cuarzo. El dispositivo puede ser programado sobre por ISP en el conector X204. El terminal MOSI de este conector también sirve para conectar un medidor análogo, el cual debería tener una deflexión de fondo de escala de 5 V.

La placa del procesador puede ser alimentada con cuatro pilas AA (recargables o secas). El diodo de la entrada de tensión protege contra una conexión accidental con inversión de polaridad y hace caer

Figura 1. La placa del sensor, dentro de la prueba, forma una sonda de prueba de temperatura con una salida digital I²C.

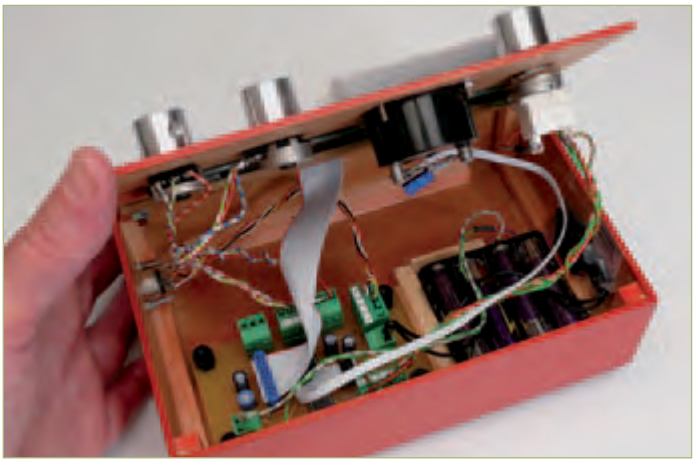


Figura 2. La placa del procesador procesa los datos I²C y controla un medidor analógico y un panel LCD.

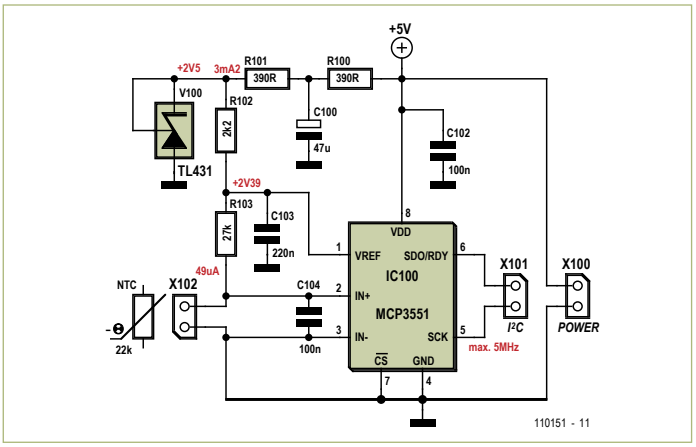


Figura 3. Esquema eléctrico de la placa del sensor.

la tensión de la batería a unos 0,6 V, asegurando que no se sobrepasa la tensión máxima de funcionamiento permisible de 5,5 V del procesador y del conversor A/D.

Los planos de las placas de circuito impreso (ver **Figura 6** y **Figura 7**) para las dos partes del circuito, están disponibles gratuitamente, tanto en archivos PDF como en archivos Protel [1].

Software

El programa interno del microprocesador también se puede descargar gratuitamente desde la página web de Elektor [1].

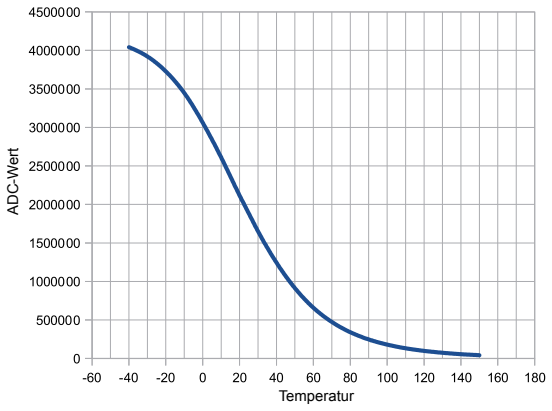


Figura 4. Curva característica del sensor.

El programa lee periódicamente los resultados de conversión de la placa del sensor y los convierte a grados centígrados (Celsius). Posteriormente, estos datos son transmitidos como caracteres ASCII por el interfaz serie (por ejemplo como '23,5341°C': señalar el uso de la coma en lugar del punto decimal, también podemos aprovechar y volver a colocar el punto decimal). La curva característica del sensor se almacena como una tabla, en incrementos de cinco grados, y los valores entre las entradas de la tabla son interpolados de forma lineal. La precisión absoluta total resultante es de, aproximadamente, dos grados sobre un rango de 0 °C a 40 °C. La mayoría del error residual es atribuible a la variación, de parte a parte, en el termistor. Si el termistor está calibrado, se puede conseguir una exactitud mayor de 0,5 grados.

Los valores de temperatura son pasados a través de una cadena de dos filtros paso bajo realizados por programa. El primer filtro elimina las variaciones que se producen en una escala corta de tiempos y el gradiente de temperaturas se calcula como la diferencia entre la entrada y la salida del segundo filtro. Los valores de tensión que sobrepasan los ajustes de los potenciómetros P1 y P2, se convierten a formato digital utilizando el conversor A/D del ATmega88. De aquí se obtienen dos constantes de tiempo, t_1 y t_2, una para cada filtro paso bajo. Los filtros funcionan como sencillas redes RC, por lo que, con un paso de tensión en la entrada, la salida se establece dentro del 99 % de su valor final, dentro de cinco constantes de tiempo. Los ajustes del potenciómetro son modificados antes de ser usados como

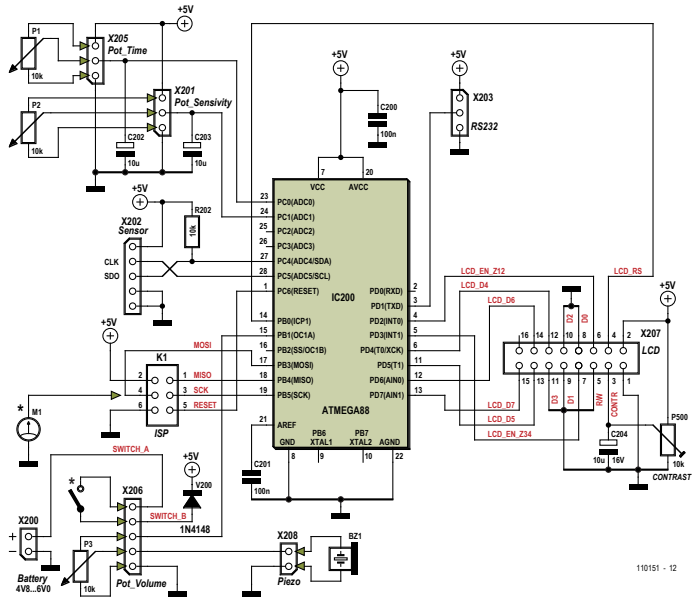


Figura 5. Esquema eléctrico de la placa del procesador.

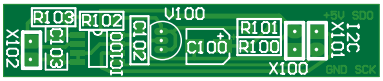


Figura 6. Plano de montaje de componentes de la placa del sensor.

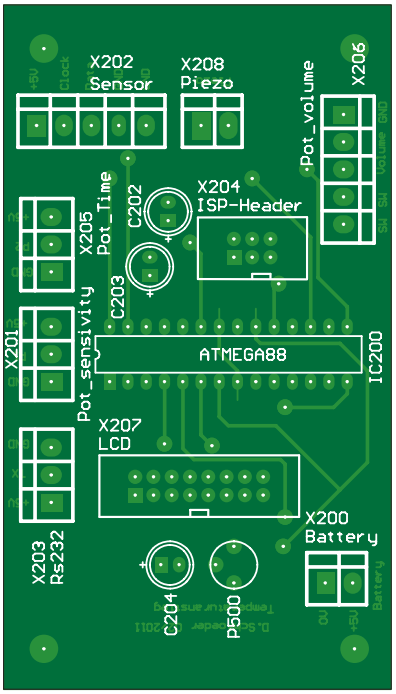


Figura 7. Plano de montaje de componentes de la placa del procesador.

constantes de tiempo a fin de permitir un ajuste más fácil sobre un amplio rango de períodos de tiempo. No hay ningún mérito en establecer la constante de tiempo del segundo filtro más corta que el del primero, por lo que la primera constante de tiempo se añade a la segunda. El rango disponible para t_1 es de 0 segundos a 19 minutos, y para t_2 es de 2 segundos a 100 minutos. Generalmente, la diferencia entre TP1 y TP2 debería ser un factor de 2. Una pequeña cantidad de histéresis evita picos ("jitters") en los ajustes.

El gradiente de temperaturas también se puede indicar sobre un medidor analógico. Para ello, se mide la tensión de la batería y la relación marca-espacio de la señal de salida PWM se ajusta para que si el gradiente es cero, la tensión media de salida sea de 2,5 V. Si el medidor tiene una deflexión de fondo de escala de 5 V, entonces señalará exactamente a la mitad de su escala.

La salida PWM es configurada de forma que la lectura a fondo de escala del medidor sea ± 0,05 °C por segundo.

Se utiliza un temporizador para crear un sonido cuyo tono depende del valor absoluto del gradiente de temperatura. Si la temperatura cae, se utiliza un segundo temporizador para interrumpir el tono periódicamente, generando un sonido intermitente en lugar de un tono continuo. Si la temperatura sólo varía dentro del nivel de ruido típico del sistema, el tono es deshabilitado.
En la pantalla LCD se muestra una tabla que presenta la temperatura, los ajustes de las constantes de tiempo y el gradiente de temperaturas calcula: (ver **Figura 8**).
La segunda línea muestra la temperatura actual con cuatro cifras decimales. El último de estos dígitos no es significativo, ya que su significado está por debajo del nivel de ruido típico del sistema. Si la sonda de prueba está bien aislada, la variación típica del último dígito es de, aproximadamente, 3 ó 4 unidades.
La tercera y la cuarta línea muestran los valores de temperatura después de ser procesados por los filtros paso bajo, con las constantes de tiempo establecidas por los potenciómetros, mostradas en la segunda columna. La última columna muestra el número en el que realmente estamos interesados: el gradiente de temperatura. El valor de la tercera fila es la diferencia entre la lectura actual y la salida del primer filtro, mientras que el valor de la cuarta fila es la diferencia entre la salida del primer filtro y la del segundo.

Funcionamiento

Inicialmente, colocaremos los potenciómetros P1 y P2 en sus extremos izquierdos de su recorrido, los cuales establecen las constantes de tiempo del filtro a 0 s y 2 s, respectivamente. Ahora es buena idea esperar un rato a que el sensor de temperatura se caliente: esto no es un efecto insignificante ya que el termistor disipa unos 0,2 mW, lo que provoca un ligero aumento de la temperatura. Seguidamente, mantendremos colocada nuestra mano a unos 10 cm del sensor durante un par de segundos y veremos que la aguja del medidor se mueve claramente a la derecha como consecuencia de la radiación térmica. Retiraremos ahora la mano y la aguja se moverá a la izquierda ya que el sensor se enfría.
El autor utiliza para la mayoría de las mediciones un valor de 5-10 s en la tercera línea del LCD (TP1) y de 20-120 s en la cuarta línea del LCD (TP2).

Figura 8. Las lecturas se presentan en una pantalla de cuatro líneas.

(110151)

Referencias en Internet

[1] www.elektor.es/110151

Acerca del autor

Dr Dietmar Schröder trabaja como desarrollador de software y hardware en Aixcom PowerSystems GmbH, en Alemania. Tiene una página web personal (en alemán) que contiene un cierto número de proyectos poco usuales en http://www.zabex.de.

Curso de audio DSP

Parte 6: generador digital de audio

Un generador de señal digital de audio para laboratorio ha de producir sinusoides con mínima distorsión, con frecuencia y amplitud ajustables para medir la respuesta en frecuencia, y tener una distorsión armónica muy baja en los dispositivos de audio y otros filtros paso bajo y banda para poder tomar medidas electroacústicas en transductores. Para estas tareas, un DSP es especialmente útil. En esta parte de la serie, la tarjeta DSP actúa como componente central en la obtención de la señal, funcionando como generador de laboratorio, permitiendo la generación de señales tanto digitales como analógicas de alta calidad.

Alexander Potchinkov (Alemania)

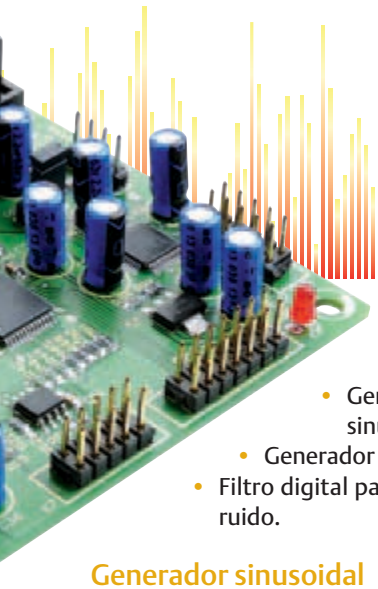
Un generador de señal pertenece al equipamiento más básico en cuanto a unidades de medida, también para pequeños laboratorios de audio o electroacústica. Resulta especialmente versátil si además de señales analógicas, también genera digitales y encima ha sido implementado con dos canales. Así podemos suministrar señales de test tanto a dispositivos analógicos como digitales, incluso al mismo tiempo. En el laboratorio y en cualquier equipo siempre hay algo para medir, tanto la respuesta en frecuencia de unos altavoces diseñados por nosotros mismos como la de unos comerciales ya listos para utilizarse en el exterior. Aparte, hemos de medir las distorsiones lineales y no lineales en los dispositivos de audio. El generador de señal descrito en esta edición dispone de señales de test o medida, cuyas características no falsearían el resultado de una medición. Cuando el ancho de banda no tiene que ser demasiado grande (para medidas de audio, en la mayoría de los casos vale con la frecuencia audible, de 20 Hz a 20 kHz), el generador puede desarrollarse mediante un DSP. Así obtenemos un precio inmejorable y a la vez calidad en las señales, con sinusoides sin distorsiones y con escaso ruido, y todas las ventajas que ofrecen los filtros. En generadores analógicos, estas características no se pueden obtener, aún en iguales condiciones. Para un generador de laboratorio hemos de completar la tarjeta DSP con un interfaz de usuario para introducir los parámetros de ajuste del generador y, si fuera necesario, algunas etapas de salida analógicas. En esta entrega presentamos el tratamiento de la señal necesario acompañado de un programa DSP y, finalmente, cómo llevar a cabo un interfaz de usuario.

Tratamiento de señales del generador digital de audio

En la **figura 1** se muestra el diagrama de bloques con dos generadores para ambos canales iguales, que los hemos llamado canal *izquierdo* y *derecho*. Uno es un generador sinusoidal de frecuencia ajustable y el otro un generador de ruido blanco. Aparte, disponemos de un banco de 42 filtros para ruido distintos, dos conmutadores multicanal para seleccionar la fuente de la señal y un atenuador. Dependiendo de la posición del conmutador y el filtro seleccionado podemos generar señales sinusoidales o bien ruido filtrado. Ya que ambos canales son independientes entre sí, es posible generar al mismo tiempo señales sinusoidales y ruido (filtrado), cada una en un canal.

El tratamiento de la señal en la figura 1 necesita cinco parámetros de valor positivo, el incremento de fase $d\phi$ para ajustar la frecuencia de la sinusoide, el índice F_I del filtro de ruido (podemos comprobar este filtro en una de las posiciones del conmutador), la posición del conmutador S_1 para seleccionar la fuente de la señal, el factor de atenuación α y el factor de ganancia ruido-señal Gain, dado en etapas de 6 dB (duplicando la tensión de salida). Jugando con los parámetros α y Gain fijamos la amplificación entre dichas etapas de 6 dB. Dado el diseño con dos canales, al final el programa DSP necesita dos sets de cinco parámetros cada uno.

En adelante, describiremos uno tras otro los tres bloques del tratamiento digital de señales,



- Generador sinusoidal,
- Generador de ruido y
- Filtro digital para la forma del ruido.

Generador sinusoidal

Para generar una señal sinusoidal del tipo $x(n) = a \sin(2\pi n f_S/f_T)$, $n = 0,1,...$ basta con una frecuencia de muestreo f_T, la frecuencia de oscilación $f_S<f_T/2$, un ángulo de fase y la amplitud a; se trata de la expresión más utilizada para obtener formas sinusoidales. Para nosotros, la técnica analógica tradicional utiliza osciladores realimentados, conocida como *puente de Wien*; la digital utiliza *oscilaciones recursivas de segundo orden* y al igual que la analógica requiere una baja distorsión de la señal y trabajar con una amplitud controlada. Entre otros, se utiliza un *acumulador de fase* con una *fuente de la señal no lineal*, los lectores de Elektor probablemente ya conozcan la técnica del DDS (Direct Digital Synthesis, Elektor de Octubre de 2003, medidor de RF). Mediante el generador digital de sinusoides, el acumulador de fase genera una señal con diente de sierra cuyo periodo coincide con el de la sinusoide. Esta formación no lineal se encarga de que la señal de diente de sierra se corresponda con la sinusoide desea. Esta técnica ya se ha utilizado en analógica mediante redes de funciones con diodos y una señal del acumulador de fase triangular, que debido a las inevitables desviaciones y la dependencia térmica de los componentes, conduce a resultados no deseados. Para las funciones no lineales de los generadores de sinusoides digitales se utiliza una llamada *Look-Up-Table*, en la que se almacena por ejemplo una señal sinusoidal muestreada mediante 1024 valores. En casos en los que el acumulador de fase tenga valores que se encuentran entre puntos de muestreo de la tabla, se interpolan linealmente tales valores. El principal inconveniente de esta técnica es que las distorsiones de la señal dependen de la frecuencia, y son mayores cuando han de obtenerse muchos valores por interpolación. Una alternativa es la aproximación polinómica, que por supuesto requiere más carga aritmética debido al cálculo de un polinomio, pero por otro lado nos ahorramos la Look-Up-Table y garantizamos la mínima distorsión posible, que de otro modo dependería de la frecuencia. Optamos por esta técnica ya que nuestro objetivo principal es la calidad del espectro de la señal. Creemos que los esfuerzos para desarrollar un tratamiento de señales sinusoidales sólo merecía la pena si se podía alcanzar la mínima distorsión, como si se tratase de un tratamiento de señales analógicas. La carga computacional antes mencionada sobre el DSP es tan baja, que esto no nos sirve como argumento para negarnos a realizar un sencillo generador de sinusoides. Para los lectores interesados, mencionar que los coeficientes de nuestro polinomio se obtienen mediante la aproximación de *Tschebyscheff*, contamos con un polinomio de grado 11 y se analiza con el esquema *Horner*.

El acumulador de fase empieza a cero, y con cada punto de muestreo su valor crece según los incrementos de fase $d\phi = 2f_S/f_T$. Con una frecuencia de muestreo de f_T = 48 kHz debería obtenerse una sinusoide de frecuencia f_S = 3 kHz, con lo que elegimos como incremento de fase $d\phi=2*3/48=0,125$. Los cálculos en complemento a dos del DSP56374, en donde podemos imaginarnos las cifras en colocadas en un anillo, nos dan la señal con diente de sierra deseada sin más, lo cual puede explicarse si consideramos el orden de dicho anillo. El acumulador de fase se calcula en doble precisión con 48 bits, permitiendo al DSP hacer "Long-Moves" de 48 bits de longitud. La estabilidad de la frecuencia en nuestra señal depende del oscilador de cuarzo montado en la tarjeta DSP, pero para aplicaciones de audio es más que suficiente.

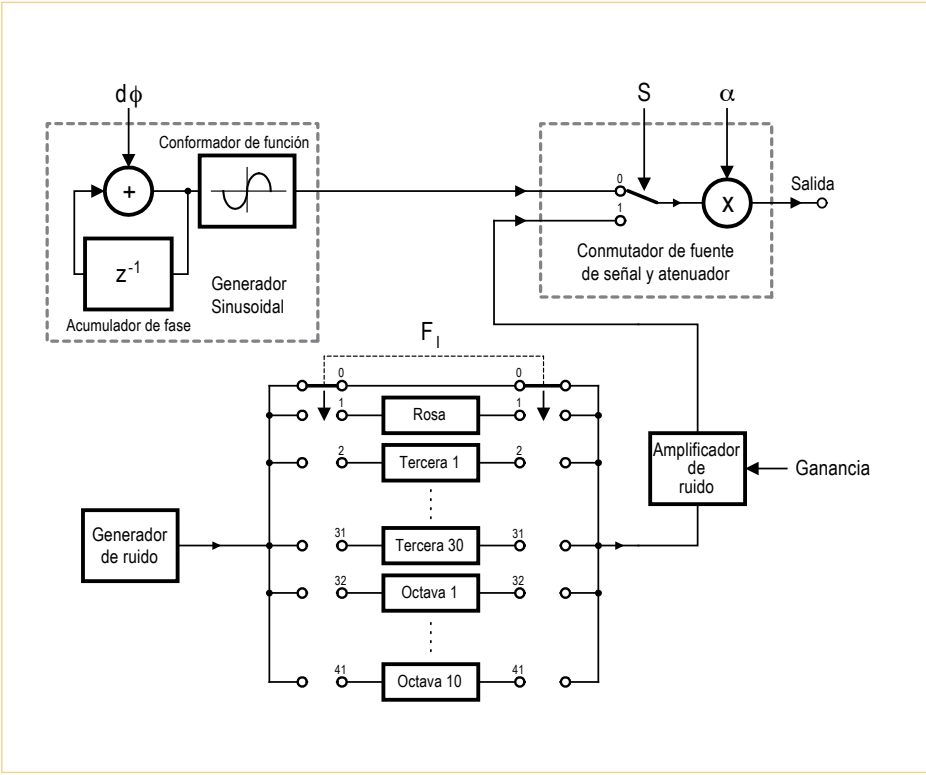


Figura 1. Diagrama de bloques del tratamiento de señales.

Generador de ruido

Incluso para un generador de ruido existen muchas opciones para implementarlo. Hemos optado por la más simple, es decir, utilizando un registro de desplazamiento con realimentación. "Sencillo", en este

Tabla 1: Índices del filtro para el switch

Índice del filtro	Filtro
0	Ruido blanco (sin forma espectral)
1	Filtro ruido rosa
2	Filtro de terceras, frecuencia media f_m = 25 Hz
...	..
31	Filtro de terceras, frecuencia media f_m = 20 kHz
32	Filtro de octavas, frecuencia media f_m = 31,5 Hz
...	..
41	Filtro de octavas, frecuencia media fm = 16 kHz

Tabla 2: Valores de la ganancia para filtros de terceras y octavas para una frecuencia media de f_m = 1 kHz

	Filtro de terceras		Filtro de octavas	
Valor de la ganancia	izquierda	derecha	izquierda	derecha
–3 dB	895 Hz	1117 Hz	718 Hz	1393 Hz
–20 dB	790 Hz	1266 Hz	509 Hz	1958 Hz
–40 dB	611 Hz	1636 Hz	291 Hz	3385 Hz
–60 dB	385 Hz	2581 Hz	145 Hz	6507 Hz

Tabla 3: Parámetros del programa y sus rangos de valores

Parámetro	Rango de valores	Tipo del dato	Tamaño	Posición
SL, SR	[0,1]	Entero	24	A la derecha
FiL, FiR	[0,1,...,41]	Entero	24	A la derecha
DphiL, DphiR	(0,1)	Fraccionario	48	A la izquierda
AlphaL, AlphaR	(0,1)	Fraccionario	24	A la izquierda
GainL, GainR	[0,1,...,6]	Entero	24	A la derecha

*Los corchetes indican que los valores límite están incluidos en el rango, mientras que los paréntesis indican que no lo están.

Tabla 4: Parámetros por defecto

Parámetros de programa del canal izquierdo		Parámetros de programa del canal derecho	
Nombre	Valor por defecto	Nombre	Valor por defecto
DphiL	0,041666667	DphiR	0,041666667
FiL	18	FiR	18
SL	0	SR	1
AlphaL	0,5	AlphaR	0,5
GainL	0	GainR	1

Tabla 5: Archivos de programa del generador de audio digital

AudioGen.asm	Programa principal
Kocz_SinCoef.tab	Coeficientes de los polinomios de la senoidal
ElektorFilter.tab	Coeficientes del filtro digital
src4392.tab	Serie de bytes para programar el SRC
ivt.asm	Entrada en la tabla de vectores de interrupción, “Audio-Interrupts”
esai4r2t.asm	Audio ISR, 4 Canales de entrada, 2 de salida
mioequ.asm	Datos útiles y aclaraciones para las direcciones IO del DSP

caso, no significa “de poco valor”. Esta técnica del registro de desplazamiento realimentado se ha utilizado ya desde hace tiempo para realizar un montaje (muy simple) de generadores de ruido analógicos con la ayuda de integrados lógicos. Una vez que el lector vea el código DSP, se convencerá de que el diseño del software va a ser más simple aún. La señal periódica generada ya es una pseudo-señal de salida, pues en cada periodo el registro de desplazamiento cambia de estado correspondientemente. Hay uno de los posibles estados (con todos los registros con el valor cero) prohibido, ya que no puede abandonar esta condición. Para dar un ejemplo numérico, supongamos un registro de desplazamiento con 4 flip-flops, del cual conocemos los 16 estados posibles, siendo 15 = 2^4-1 los estados permitidos (correspondientemente los números del 1 al 15). Utilizamos dos generadores con registros de desplazamiento, cada uno con 24 flip-flops, con lo que tenemos un periodo cuya duración es de siete minutos y con una frecuencia de muestreo de 48 kHz. Para cualquier aplicación de audio esto es más que suficiente. Ambos generadores utilizan diferentes configuraciones en la realimentación, lo que significa que en la práctica están lo bastante descoordinados, es decir, estadísticamente proporcionan señales totalmente independientes. Por ello, pueden tomarse medidas en dos canales reales. A este respecto, una vez más remarcamos la prioridad de la calidad del tratamiento de la señal en este proyecto.

Filtros

Los filtros establecen configuraciones espectrales de ruido determinadas, y en el banco completo se incluyen 42 de ellos, cuyos índices se especifican en la **tabla 1**:

- Un filtro dummy para el ruido blanco.
- Un filtro de ruido rosa, con el cual puede obtenerse este ruido a partir del blanco mediante un filtrado de paso bajo. Entre los lectores, los expertos en altavoces saben que este ruido no sólo protege los tweeters de posibles sobrecargas y el sobrecalentamiento, sino que si chequeamos su señal con un analizador veremos que los niveles de las terceras octavas son (idealmente) iguales.

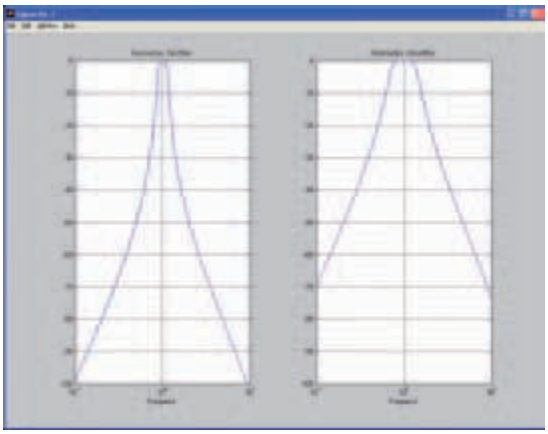


Figura 2. Frecuencias portadoras de un filtro de terceras y octavas para frecuencias medias.

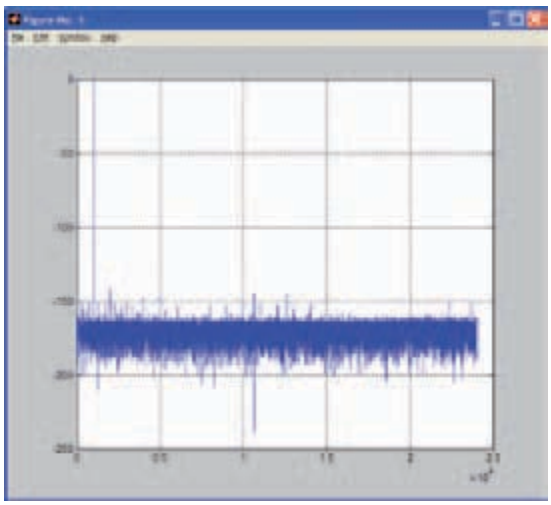


Figura 3. Espectro de una señal sinusoidal de 1 kHz obtenida digitalmente.

- 30 filtros de terceras para generar ruido de las bandas de terceras octavas con las frecuencias medias de 25, 31.5, 40, 50, 63, 80, 100, 125, 160, 200, 250, 315, 400, 500, 630, 800, 1000, 1250, 1600, 2000, 2500, 3150, 4000, 5000, 6300, 8000, 10000, 12500, 16000 y 20000, todas dadas en hertzios según la normativa DIN EN 61260.
- 10 filtros de octavas para generar ruido de las bandas de octavas con las frecuencias medias de 31.5, 63, 125, 250, 500, 1000, 2000, 4000, 8000 y 16000, todas dadas en hertzios según la normativa DIN EN 61260.

La **figura 2** ejemplifica las respuestas de la frecuencia portadora ante un filtro de terceras octavas y uno de octavas a frecuencia normalizada. Si tomamos la frecuencia media del filtro de 1 kHz en el medio acústico, en la **tabla 2** podremos ver los valores de las respuestas en frecuencia según las medidas de la ganancia. En dicha tabla se han marcado las frecuencias de corte a 3 dB.

Ya que se trata de un filtro digital y no utilizamos una frecuencia de muestreo demasiado alta (48 kHz), el filtro se desvía algo a la derecha de la frecuencia media de la entrada analógica en las altas frecuencias. La razón es que estamos a la mitad de la frecuencia de muestreo, o sea, 24 kHz, con lo que tenemos atenuación infinita. En el tratamiento de señales a esto se le llama punto de *cero en la respuesta en frecuencia*. Se pierde la simetría entre la derecha y la izquierda, pero sin embargo mejora la respuesta en el eje derecho, con lo que este efecto no es un inconveniente del todo.

Todo filtro, tanto los *recursivos* como los *IIR* de grado 6, que pueden realizarse también de forma analógica, parten siempre de 3 filtros parciales de grado 2 colocados uno tras otro. Un filtro de grado 6 necesita una estructura determinada con una memoria para almacenar sus estados, dotada con 8 registros. Estos registros de estado se corresponden con los condensadores y/o inductancias de los filtros analógicos. En otros casos, un filtro necesita 15 coeficientes de filtrado para establecer sus características. Hemos almacenado los 41*15=615 coeficientes de los filtros en una tabla del DSP. Por motivos de programación, hay un filtro nº 42 que no tiene función específica y se utiliza sólo para el ruido blanco. Dicho filtro tiene el índice 0.

Si queremos cambiar un filtro, sólo tendremos que señalar con el puntero a los coeficientes de filtro deseados. Esto puede calcularse fácilmente. Partiendo de la dirección base de coeficientes A_B (comienzo de la tabla) y el índice del filtro F_I se calcula la dirección del puntero $A = A_B + 15 * F$, la cual indica el origen de un conjunto de coeficientes del filtro. Aquí vemos uno de los puntos fuertes del tratamiento digital de señales. Tan sólo imaginemos lo que hubiera costado implementar esto con bancos de filtros analógicos. El autor tiene en su laboratorio un viejo banco de filtros de terceras octavas y octavas de Brüel&Kjaer, que fácilmente pesará 15 kg. Sin embargo, nosotros hemos concentrado cientos de filtros en la memoria del DSP, de una forma tan simple como es “orientar los punteros”.

Para los lectores interesados, diremos que 40 de los filtros paso banda se corresponden con las especificaciones de clase 0 de la normativa DIN EN 61260. El filtro de ruido rosa tiene entre los 10 Hz y los 20 kHz una desviación inferior a 0,1 dB. Los filtros paso banda pueden utilizarse sin distorsiones hasta los 100 dB, lo cual es difícilmente posible utilizando sus homónimos analógicos.

Programa DSP

El programa DSP para el tratamiento de la señal se ha implementado como un bloque en el audioloop (o bucle de audio). Incluso si no leemos señales de audio en el generador, en este paso no modificamos el curso de audioloop y leemos las señales de audio “proforma”, debido a la sincronización, y porque queremos utilizar para todos los proyectos el mismo archivo base. El programa DSP utiliza diez parámetros de programa como parámetros locales. La **tabla 3** enumera los parámetros, sus valores y otras características.

Los valores por defecto se han elegido de modo que el canal izquierdo genera una sinusoide de frecuencia 1 kHz y el canal derecho ruido de bandas de terceras octavas con una frecuencia media de 1 kHz. En

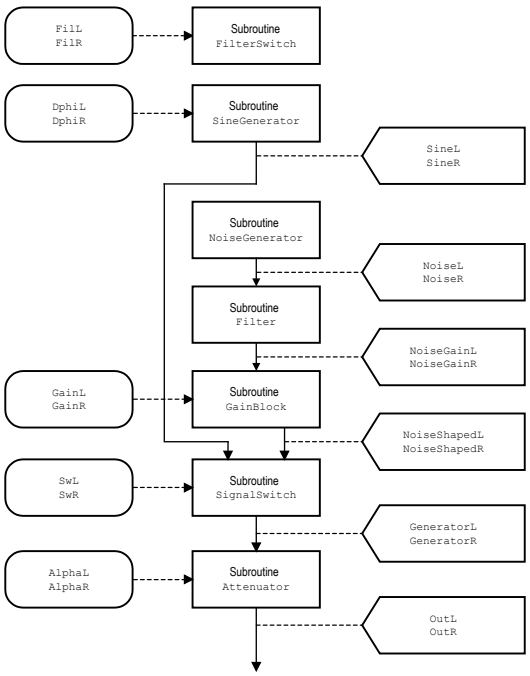


Figura 4. Subrutinas y señales en el audioloop.

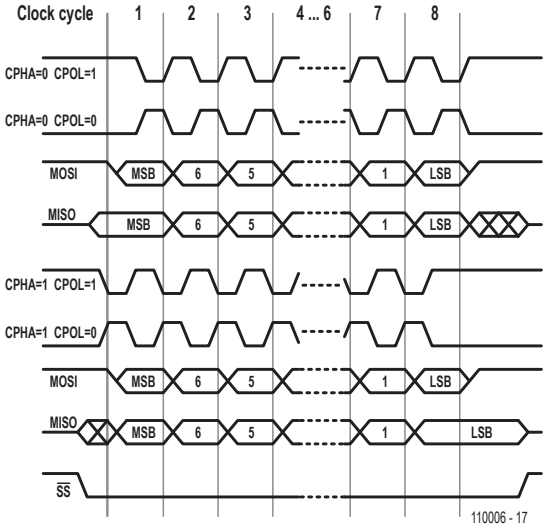


Figura 5. Temporización del puerto SPI, fuente de la imagen: Wikipedia.

ambos casos, el atenuador divide los valores de las señales o limita sus niveles de salida en 6 dB. La etapa amplificador ruido-señal de ambos canales está fijada a 0 dB. Los parámetros por defecto se almacenan en la tabla 4. Los archivos necesarios para el programa DSP se enumeran en la tabla 5.

Subprogramas y señales en el audioloop

Dos subprogramas se implementan en la rutina del audioloop. La subrutina `InitStateVars` borra la memoria de estado del filtro y ambos acumuladores de fase. El estado de los registros de desplazamiento se fija mediante un valor determinado. Con la subrutina `SetDefaultParams` se fijan los parámetros acorde con la **tabla 4**.

Dentro del audioloop hay siete subrutinas más, de las cuales seis participan en el tratamiento de la señal de audio. El subprograma `FilterSwitch` lee ambos parámetros del filtro `FiL` y `FiR` y calcula las direcciones de inicio correctas para los juegos de coeficientes de filtros correspondientes, cuya longitud es de 15. Esta rutina no tiene nada que ver con el tratamiento de señales, sino que es más bien una conversión de parámetros. Se ejecuta en el audioloop, ya que los parámetros tienen que poder cambiarse por la marcha. Aparte, ya que han de calcularse las direcciones correctas y el resto de parámetros tienen que poder seguir siendo utilizados, tal cálculo ha de ejecutarse dentro.

La subrutina `SineGenerator` contiene dos generadores independientes para los canales izquierdo y derecho. Ambos parámetros de incrementos de fase `DphiL` y `DphiR` establecen la frecuencia de oscilación. El verdadero cálculo de la función sinusoidal mediante aproximación polinómica se ejecuta en la macro `sine`. Las variables de estado de la subrutina son los registros acumuladores de fase `l:PhaseAccuL` y `l:PhaseAccuR`, en donde se almacenan los valores actuales de las fases. Aparte se necesitan los seis coeficientes necesarios para la aproximación por polinomios, que se guardan en la Y-RAM en la dirección base `SinCoef`. Ambas señales sinusoidales pueden verse en `y:SineL` e `y:SineR`.

La subrutina `NoiseGenerator` incluye los cálculos estadísticos de los dos generadores de ruido independientes para cada canal, izquiero y derecho. También se utiliza una macro para la generación de ruido. La macro utiliza dos argumentos, un espacio del registro de desplazamiento y una constante de 24 bits para ajustar la realimentación. Las variables de estado son los dos registros de desplazamiento de 24 bits `y:NoiseL` e `y:NoiseR`, que también son las señales de salida de los generadores de ruido.

El subprograma `Filter` incluye el filtro de ruido. Se trata de un filtro *recursivo* o *IIR* de orden 6. Para su implementación, se utiliza la macro `iir2mac` de la librería de programas de Freescale, anteriormente Motorola. La macro puede utilizarse para filtros de orden arbitrario, pues se trata de un programa estándar. Esta subrutina no necesita parámetros, ya que las direcciones base de ambos juegos de coeficientes de longitud 15 ya se calcularon en la subrutina `FilterSwitch`. Necesita dos memorias de estado, cada una con ocho registros, cuyas direcciones base sean `FstateL` y `FstateR`. Ambas señales de ruido filtrado pueden verse en `y:NoiseGainL` e `y:NoiseGainR`.

Filtro de octavas y ganancia ruido-señal

Con nuestro generador, podemos obtener ruido de terceras y octavas a partir de ruido blanco. Si para el ruido blanco tomamos un ancho de banda de 0 a 24 kHz con L_R=0dB, el nivel del ruido filtrado es bajo, ya que por el filtro sólo está pasando parte del ruido. Veámoslo más de cerca abajo. Los anchos de banda B (distancia entre la frecuencia inferior y la superior) para filtros ideales es función de f_m

Filtro de terceras $B = (2^{1/6} - 2^{-1/6}) * f_m = 0{,}2316 * f_m$

Filtro de octavas $B = (2^{1/2} - 2^{-1/2}) * f_m = 0{,}7071 * f_m$.

Con estos anchos de banda podemos calcular los niveles de potencia en decibelios:

Nivel del filtro de terceras $L_{Terz,fm} = L_R + 10*\log_{10}(f_m) + 10*\log_{10}(0{,}2316/24000) = L_R + 10*\log_{10}(f_m) - 50{,}1547$

Nivel del filtro de octavas $L_{Oktav,fm} = L_R + 10*\log_{10}(f_m) + 10*\log_{10}(0{,}7071/24000) = L_R + 10*\log_{10}(f_m) - 45{,}3073$.

Resumimos los resultados en la siguiente tabla:

Nivel del ruido filtrado		
Frecuencia media f_m	Nivel del filtro de terceras $L_{Terz,fm}$	Nivel del filtro de octavas $L_{Oktav,fm}$
25	-36,18	-30,32
31,5	-35,17	
40	-34,13	
50	-33.17	-27,31
63	-32,16	
80	-31,12	
100	-30,15	-24,34
125	-29,19	
160	-28,11	
200	-27,14	-21,33
250	-26,18	
315	-25,17	
400	-24,13	-18,32
500	-23,17	
630	-22,16	
800	-21,12	-15,31
1000	-20,15	
1250	-19,19	
1600	-18,11	-12,3
2000	-17.14	
2500	-16,18	
3150	-15,17	-9,29
4000	-14,13	
5000	-13,17	
6300	-12,16	-6,28
8000	-11,12	
10000	-10,15	
12500	-9,19	-3,27
16000	-8,11	
20000	-7,14	

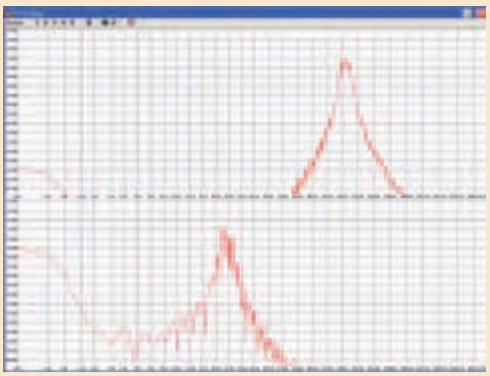


Figura A. GainL = 2, GainR = 3, sin saturación.

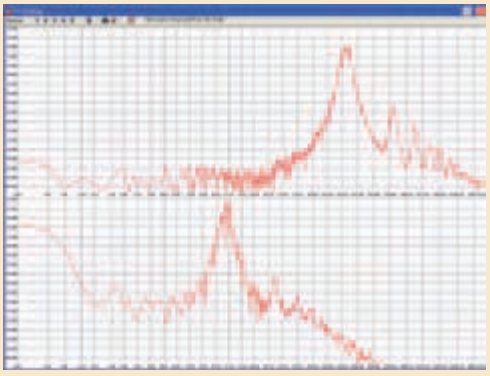


Figura B. Saturación, distorsión no lineal y límite de la señal, GainL = 3, GainR = 5.

Resulta fácil percatarnos de que por cada tercera tenemos una atenuación de 1 dB, y con el filtro de octavas de 3 dB por cada una. Si necesitamos más nivel para las bajas y medias frecuencias podemos amplificar las señales mediante los parámetros `GainL` y `GainR` en pasos de 6 dB. Pero si lo compensamos incrementando el nivel justo exactamente, el generador saturará, lo cual dará un espectro erróneo; hemos de elegir valores de ganancia de modo que los picos no coincidan nunca. Sirvan como guía los valores de la tabla. No obstante, también debemos analizar caso a caso, tanto el espectro como la señal. Las **figuras A y B** muestran el efecto de la saturación en el ruido de terceras. Con nuestro generador obtenemos ruido de terceras en dos canales, en el izquierdo con una frecuencia media de f_m = 1 kHz y en el derecho con f_m = 80 Hz. En la figura A, generada con Wavelab, hemos elegido una ganancia de 12 dB para la izquierda y de 18 dB para la derecha. En la figura B correspondientemente 18 dB y 30 dB. En esta imagen puede distinguirse bien parte de la distorsión, cuyo origen está en los niveles de la señal, apareciendo saturación. Si desea utilizar una alta ganancia, primero deberemos ver con un analizador de espectro si aparece saturación o no.

La subrutina `GainBlock` incluye la etapa de amplificación para las señales de ruido filtradas. La ganancia no está limitada y por lo tanto sólo debe utilizarse si la señal filtrada tiene una modulación demasiado baja. Los detalles al respecto se encuentran en una sección adicional. El diseño del programa es realmente simple. Se implementa una ganancia de 6 dB, (se duplican los valores de las muestras) y se aplica un desplazamiento a la izquierda. Ya que el DSP es por definición multidesplazamiento, puede realizar muchos avances de un solo paso hasta conseguir los 6 dB de ganancia. Ambas señales de ruido filtradas y amplificadas pueden chequearse en `y:NoiseShapedL` e `y:NoiseShapedR`.

La subrutina `SignalSwitch` incluye el conmutador de la fuente de la señal, con el cual alternamos entre señales sinusoidales o de ruido. El conmutador funciona con dos parámetros, `SwL` y `SwR`. No son necesarias variables de estado ni coeficientes. Las señales seleccionadas mediante el conmutador pueden verse en `y:GeneratorL` e `y:GeneratorR`.

El último subprograma del audioloop es `Attenuator`. Aquí se calcula la atenuación directamente en la señal de salida. Los factores de atenuación se ajustan mediante dos parámetros, `AlphaL` y `AlphaR`. Tampoco se necesitan ni variables de estado ni coeficientes. Las señales atenuadas quedan a disposición en `y:OutL` e `y:OutR` y se transfieren a la salida de audio.

Ajuste de parámetros

Para poder cambiar las señales del generador de audio en marcha, hemos de cambiar dos veces cinco parámetros para cada canal. Actualmente esto tiene que hacerse en el programa DSP, en la subrutina `SetDefaultParams` que incluye los parámetros deseados, ensamblando de nuevo el programa y cargándolo en el DSP. Sería mucho más elegante poder introducirlos mediante un interfaz con display y escribir los parámetros vía serie, SPI, en el DSP.

También es posible conectar el DSP a un teclado o un potenciómetro digital con LCD, pero esto ya se escapa de lo que representa el tratamiento de señales en sí. Esto debería mantenerse separado, utilizando un microcontrolador para la entrada de valores. No nos vamos a poner a recomendar controladores, ya que hay una gran variedad. También puede programarse en su propio lenguaje ensamblador o en otro (cómodo) lenguaje de alto nivel. El autor utilizó para esto un microcontrolador veterano del tipo 68HC11, que probablemente los lectores no quieran implementar, incluso a pesar de las facilidades que da su arquitectura para trabajar con DSPs.

Para utilizar el puerto SPI del DSP, **y funcionar como master**, hemos de configurar ambos registros de periféricos HCSR y HCKR. Esto significa ajustar el reloj, la polaridad del reloj y la fase, así como la longitud de la palabra (normalmente 8 bits). La **figura 5** muestra las variaciones en la temporización. Después se programa una subrutina de ajuste de parámetros, como por ejemplo una ISR a la que se llama por IRQ, que el micro sólo dispara si se están utilizando las IRQC del DSP en nuestra tarjeta DSP. El µC estará funcionando como **slave SPI**. Una alternativa a utilizar interrupciones externas (poco elegante, aunque sencilla), es preguntarle al DSP periódicamente los valores desde el microcontrolador y comprobar si hay cambios. Para la temporización puede utilizarse un múltiplo del reloj. Tenemos que probar los dos programas de ejemplo de la 4ª entrega, en los que el DSP generaba señales sinusoidales de prueba, con un contador de reloj de 192 valores. El siguiente segmento de código DSP muestra la comunicación bidireccional DSP-SPI por polling. En principio podemos ver al menos todo lo necesario, las configuraciones que hay que aplicar.

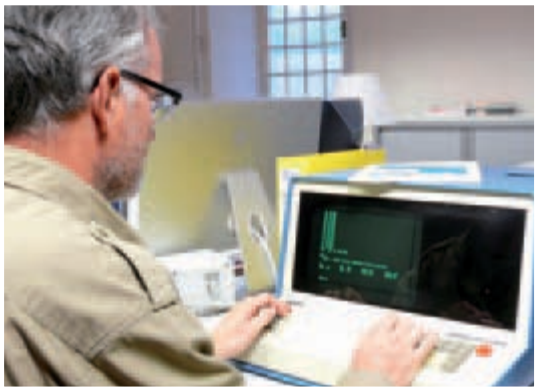
```
bclr    #HEN,x:HCSR         ; SHI disable, SPI-Reset
movep   #$002048,x:HCKR     ; Cpol=0, Cpha=0, narrow spike filter,
                            ; f=Fosc / 2 / 8 / 10 = 0.9216MHhz
movep   #$000040,x:HCSR     ; 8-Bit, Master-Betrieb, FIFO off
bset    #HEN,x:HCSR         ; SHI enable

...

move    #Buffer,r0
do   #N,RW_MuC
     jclr    #HTDE,x:HCSR,*  ; transmit register empty?
     movep   x:(r0),x:HTX
     jclr    #HRNE,x:HCSR,*  ; receive register full?
     movep   x:HRX,y:(r0)+
RW_MuC
```

El segmento de programa escribe y lee dos buffers de longitud N, un buffer de salida en la X-RAM y uno de entrada en la Y-RAM, ambos en el mismo rango de direcciones. En las primeras cuatro líneas de programa se ha implementado, configurado y activado el estado de reset. El resto de segmentos de programa giran en torno a la lectura y escritura de ambos buffers. Para utilizar el generador de sinusoides, utilizamos el incremento de fase $d\phi = 2f_S/f_T$ como valor decimal justificado a la izquierda. Sería mucho más elegante servirnos de la frecuencia de la señal f_S como parámetro entero justificado a la derecha, y que el DSP calculase por su cuenta el incremento de fase necesario. La elección de estos parámetros está en relación directa con la del microcontrolador y la del interfaz que utilicemos.

(110006)

Sistema de Desarrollo Cosmac IV

Hola Mundo desde la Edad Embedrock



Jan Buiting (Editor UK y US de Elektor)

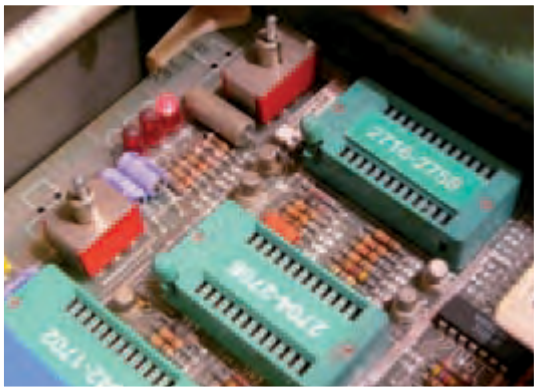
En lo que se refiere a microprocesadores, mis raíces están creadas a primeros de los años 80, cuando cualquier tipo de "sistema de aficionados" estaba basado en dispositivos de computación como el Z80, el 6502 y el 8085. Siempre he odiado la electrónica que consume potencia sin hacer nada útil, por lo que abandoné esos microprocesadores NMOS hambrientos de energía, sus clubs de fans y sus BBS y me encaminen al microcontrolador de 8 bits menos popular, dedicado al procesamiento, llamado CDP1802, originalmente diseñado y producido por RCA, la compañía que estaba detrás de la famosa serie de circuitos integrados lógicos CD4000. El 1802 CMOS (de ahí su sorprendente potencia amigable) y su familia de circuitos integrados periféricos "Cosmac", lo hicieron realmente bien en USA después del montaje de un sistema doméstico llamado Cosmac ELF, bastante bonito, que fue publicado en la conocida revista *Popular Electronics*, años atrás en 1976 (era aproximadamente el momento en que Jobs & The Woz jugueteaban con el 6502 en su garaje de Palo Alto). Una variante europea de este montaje, más sofisticada, llamada "Cosmicos", fue desarrollada unos cuatro años después y esto dio la bienvenida a mi "plataforma" personal (ver "CDP1802 - el primer micro en el espacio", Elektor, Octubre de 2006).

En 1980, como estudiante, si podías poner tus manos sobre un libro de datos ("data book") Cosmac de RCA, eras el rey. Conseguí que me dejasen prestado (de manera permanente) uno de estos libros de un hombre bondadoso en Vekano, un ex-distribuidor RCA en Holanda. Probablemente, para rellenar el libro y mantener a los comerciales contentos, las últimas 40 páginas del libro mostraban algunas de las herramientas software de RCA, circuitos, programadores, compiladores (como BASIC y PL/M) y sistemas de desarrollo Cosmac. Aunque mi propio montaje del sistema Cosmicos estaba bastante bien equipado (48 KB de memoria RAM y todo eso), era "verde" (por debajo de 200 mA con 5 V) y también rápido (3,58 MHz), me maravillaba al contemplar las especificaciones y fotografías (bastante pobres) de la principal línea de productos de promoción de RCA: el monumental Sistema de Desarrollo Cosmac IV CDP18S008. No dieron ningún precio, se suponía que tenías que telefonear.

Dependiendo de lo aficionado que fueses y de tu interés, nos puede haber llevado hasta 30 años el haber podido comprar aquello con lo que soñábamos cuando éramos jóvenes o estudiantes. Dos fuerzas juegan a nuestro favor, lentas pero seguras: (1) ganamos dinero en lugar de gastarlo y (2) el precio del objeto tan deseado ha caído hasta el nivel de basura tecnológica que nadie quiere. ¡Siempre he apreciado mi sistema Cosmicos CDP1802 y todas sus tarjetas y periféricos y, hace algunos años, mientras miraba las páginas de ELF en Yahoogroups, no podía creer lo que veían mis ojos! Un correo de un compatriota que preguntaba amablemente, a una audiencia mayormente americana, dentro del foro, si alguien estaba interesado en un sistema Cosmac IV completo. Todos rechazaron amablemente la petición debido al coste colosal de conseguir el transporte del equipo para EE.UU., desde donde había venido en 1981. Para acortar conseguí hacerme con el sistema completo, pague un precio simbólico y me lo llevé a casa. También estaba incluida una gran cantidad de circuitos adicionales realmente desarrollados con el sistema, así como documentación en carpetas y programas en discos extraíbles de ocho pulgadas. Fue la primera vez que tuve que ajustar los faros de mi coche para evitar deslumbrar al tráfico del sentido contrario. Nunca había llegado a pensar que el sistema que había visto en un libro, 30 años antes, fuese tan voluminoso y pesado. Pero, ¡qué hallazgo! La unidad azul brillante y blanco mate, llamada "Cosmac IV" es, básicamente, un terminal mudo CRT que hablaba internamente con un sistema CDP1802. Sólo pesaba 17 kilos. Podéis creerlo o no, pero el "terminal" es en sí mismo un sistema de video CDP1802, colocado entre un teclado y una pantalla CRT de 12 pulgadas. El sistema de desarrollo real es una placa de interconexión posterior sobre la que se conectan "micro-módulos" CDP18Sxx de RCA como una CPU (CDP1802), ROM, RAM, E/S, FDISK, etc. El programa que tenemos que desarrollar para una aplicación personalizada se escribe, prueba y depura completamente usando la "card nest" ("nido de tarjetas") hasta que creamos que el código hexadecimal está listo para ser volcado de forma segura en memorias (E)PROMs como la 2708 o la 2706. El programador de memorias (E)PROM está accesible bajo un panel abatible. El panel se abre y se cierra pulsando sobre él. La gente lo encuentra divertido. Sobre la parte trasera de la caja del terminal se encuentran los conectores para la alimentación de alterna AC, el disco, la impresora, el CRT EIA, SYS, EIA, MOPS EIA, reserva #1 y reserva #2.

La unidad terminal Cosmac IV tiene una nostálgica pantalla CRT en verde fosforescente, que es perfecta para leer, aunque sus 24 líneas de 80 caracteres son claustrofóbicas comparadas con las pantallas LCD de hoy en día. También es una delicia escuchar el teclado ASCII Sperry de 73 teclas cuando escribimos sobre él a gran velocidad con teclas que son del tipo efecto "Hall". Usando el editor de pantalla completa (FSE) para escribir nuestro código (en ensamblador, por supuesto) ¡apenas

RCA (CDP18S008) (1978)



echamos de menos la facilidad y el confort de un ratón, los menús, los iconos o la distracción constante de Internet!

Si pensamos que el terminal CRT azul y blanco del Cosmac IV es "caro y completo" y que no podemos esperar a comenzar a escribir el código y programar nuestras memorias ROMs, ¡espere! Aún hay dos "componentes" más que forman parte del sistema CDP18S008.

El primero se trata de una unidad de discos flexibles de ocho pulgadas, dual, con la enorme capacidad de 256 KB (0,025 GB) cada una. Fabricada por Pertec (Chatsworth & Irvine, CA), con la tapa de su caja pintada de "azul RCA", de acuerdo con el panfleto que encontré, este monstruo llamado Model 3712 pesa 34 kgs. La misma unidad de disco, pero con un color diferente, era suministrada por Altair para sus sistemas basados en el 8080 y muy similar al de otros fabricantes de mini-ordenadores. La unidad produce una horrorosa cantidad de ruido debido al enorme ventilador extractor y a los dos motores de los lectores de disco que están funcionando todo el tiempo (probablemente para mantener los tiempos de acceso al disco dentro de límites). Se escucha un pesado "clac" cada vez que una unidad de disco es seleccionada o liberada. El zumbido de las cabezas que se desplazan a lo largo de la superficie del disco hace un agradable ruido de tono de llamada. Dentro de la caja se puede ver una placa de control de casi el tamaño de dos iMAcs, "poblada" con unos 150 circuitos integrados viejos (la mayoría de ellos TTL que trabajan con 5 V). Aunque ruidosa, calurosa y lenta, como tenía que ser, la unidad de disco dual seguía funcionando fiablemente después de 30 años. Todos los discos de ocho pulgadas que conseguí de su anterior propietario pudieron ser leídos sin problemas y ahora estoy considerando el pedir un conversor de disco de ocho pulgadas a USB al Servicio de Solicitud de Conversión.

El segundo componente del sistema es llamado "Micro-monitor" (CDP18S030), donde la palabra "micro" hace referencia a un microcontrolador, no a un tamaño. Se trata de una maleta de aluminio, de nuevo pintada en azul RCA, que contiene poco más que unos LEDs, unos conmutadores y unos zócalos ZIF. La idea era la de "migrar" la CPU desde la aplicación del cliente hacia el Micromonitor (a través de un cable plano de 40 hilos) y, a continuación, ¡observar paso a paso el programa para ver lo que las líneas de la CPU estaban haciendo! Así pues, la maleta azul, que no era la adecuada para un vendedor ambulante que apareciese en *The Jetsons*, fue diseñada para ayudar a depurar las aplicaciones CDP1802 "en el sistema, en tiempo real e in-situ". Pero, ¿justo ahora? Realmente no podía entenderlo ya que pronto se necesitaba ver el código hexadecimal y, posiblemente, entrar en el de un modo confortable. Una versión posterior (CDP18S030A) tenía una unidad de pantalla/teclado desmontable que se parecía a una calculadora de bolsillo de los años 70.

Los únicos circuitos de aplicación CDP1802 producidos en masa que parecen haber sobrevivido han sido un sistema de semáforos fabricado en EE.UU. que, ocasionalmente, se ofrece en Ebay; y un radioteléfono en UHF, de Nokia, del año 1996 (!), que fue copiado y convertido para el uso de los radioaficionados. ¡Ah!, y no debo olvidar un ordenador doméstico/de juegos, llamado COMX35.

Hoy, un pequeño grupo de gente continúa disfrutando de trabajar con la CPU "Cosmac" CDP1802. Los podemos encontrar dentro de las comunidades ELF en Internet. Personalmente, uso el sistema Cosmac IV de forma ocasional para retocar el programa de mi invernadero casero y el control de riego basado en el bueno y viejo CDP1802. A menudo uso cosas como PL/M, CDOS, UT5, MOPS, BASIC1 y ASM8. Las enormes unidades de disco y el Micromonitor ya no los uso, las primeras están siendo emuladas por dos memorias RAM SMD estáticas ¡con pila de almacenamiento! Seguro que puedo apreciar 30 años de progreso y miniaturización que hemos conseguido tanto en sistemas microcontroladores como en componentes. La verdad es que el valor de los 70 kilos del CDP18S008 podrían caber fácilmente en, por ejemplo, una sencilla FPGA "Spartan" de Xilinx. Ya lo sé, mi sistema Cosmac V realmente pertenece a un museo de ordenadores.

En Internet, en 1976 se mencionó en algún sitio un precio de 70.000 $ para un sistema completo CDP18S008 como el que hemos descrito aquí. No tengo ninguna manera de verificarlo. Todo lo que sé es que el anterior propietario nunca llegó a recuperar el coste de su sistema haciendo desarrollos, vendiendo y dando soporte sobre aplicaciones altamente especializada, encargadas por clientes, incluyendo instituciones del gobierno. Esto te hace pensar por qué los sistemas de desarrollo de microcontroladores actuales ofrecidos por los fabricantes están disponibles a precios irrisorios.

Si algún lector de la sección de "Retrónica" tiene cualquier disco original Cosmac CDP18Sxx RCA que contenga herramientas de programación o lenguajes de programación de alto nivel, por favor, hacédmelo saber. Os pido lo mismo para un Microterminal CDP18S021 trabajando sobre UT5.

(110528)

Retrónica (Recuerdos de electrónica) es una columna mensual que cubre equipos electrónicos antiguos, incluyendo diseños legendarios de Elektor. Se agradecen contribuciones, sugerencias y peticiones; por favor, enviad un correo electrónico (email) a redacción@elektor.es