[Analógico … trolladores & Embebido • Audio • Test & Medida]

Curso DSP continuación

Junio 2011 N.º 372 6,50 €

+ Protocolo USB desmitificado

elektor

www.elektor.es

Detector de Radiación

Medida de los Rayos Gamma con un Fotodiodo

Desarrollando Apps para Android

Con PC, placa Beagle, teléfono o tablet

+ **Retrónica:**
La destacable prehistoria del LED

+ **Flowcode RC5:**
Añade control remoto a tus proyectos

ISSN 0211-397X
00372
9 770211 397008

Medidor de radiación para pobres

Tras el terrible accidente en la planta nuclear de Fukushima ha resurgido la demanda de contadores Geiger. No es de extrañar que muchas personas quieran establecer los niveles de radiación en su entorno. Una compañía como Conrad Electronics con varios tipos de contadores Geiger en su catálogo vio desaparecer su stock en pocos días. Las dificultades del fabricante de los dispositivos para abastecer la demanda quedan de manifiesto en la web de Conrad Electronics donde se avisa que el suministro se puede retrasar hasta ¡Febrero de 2011!

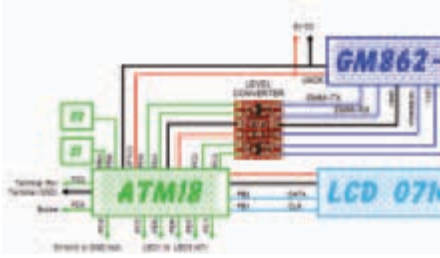
Para Elektor el repentino interés en los contadores Geiger fue razón suficiente para empezar a pensar en una detector de radiación sencillo, barato, que se pueda hacer en casa y válido para este tipo de medidas. Después de todo, la radiación es permanente y está por todas partes, y las posibles fuentes deben ser identificadas cuidadosamente, así como los niveles por ellas emitidos.

Es sabido desde hace tiempo el hecho de que un simple fotodiodo puede ser utilizado como detector de radiación si se usa adecuadamente. Cuando se combina con un amplificador de bajo ruido, tienes un instrumento básico para detectar la presencia de rayos X y rayos gamma de forma visible y audible. Aunque no es válido para medir valores absolutos, puedes tener una idea razonable de los campos de radiación que te rodean. Así que asegúrate de leer y asimilar el articulo “Fotodiodo para medir rayos gamma” en la página 34.

Otro tema candente, pero en un sentido ligeramente diferente, es el sistema operativo Android que se ejecuta en numeroso teléfonos inteligentes y otros dispositivos portátiles. En esta edición te mostramos como desarrollar Apps Android para tus propias aplicaciones de una forma relativamente sencilla. Buena parte está dedicada al hardware y software relevantes. ¡Muy recomendable también!

Diviértete con estos y los muchos otros artículos de edición de Junio de 2011. También estoy ocupado ahora editando y preparando los artículos de la próxima edición ‘Generador de Proyectos’ de Julio y Agosto.

Eduardo Corral, Editor



SUMARIO

Volume 32
Junio 2011
nº 372

28 Desarrollar para Android

Las estimaciones actuales indican que cada año salen al aire 350.000 nuevos teléfonos Android. ¿Cuál es el motivo de este asombroso suceso?¿Es Google?¿Es el hecho de que sea Código Abierto?¿Es porque funciona bien? En cualquier caso, la razón específica no es importante; lo que importa es que tú también puedes sumarte a esta tendencia y desarrollar tus propias Apps Android. ¡He aquí cómo!

34 Fotodiodo para medir rayos gamma

Los tubos contadores Geiger-Müller son caros y cada vez más difíciles de encontrar, e incluso si te las arreglas para conseguir uno, aún necesitas encontrar la forma de generar su tensión de funcionamiento de varios cientos de voltios. Es menos conocido que incluso un simple fotodiodo como el BPW34 se puede utilizar para detectar rayos X y radiación gamma.

48 Dentro del USB

El interfaz USB es sin duda alguna el estándar indiscutible de comunicación entre el PC y los dispositivos periféricos. Su mayor virtud es que resulta muy fácil de manejar para el usuario final, basta con conectar el cable para que todo funcione. ¿Cómo funciona este protocolo?

62 Seguimiento GPS con ATM18

Un módulo modem GSM con receptor GPS incorporado nos permite determinar su posición con mucha precisión. Si lo combinas con un módulo ATM18 de Elektor e instalas el lote en tu coche, tienes un dispositivo de seguimiento que puede enviarte un e-mail o un mensaje de texto para hacerte saber donde está exactamente tu precioso vehículo.

PREMO incluye antenas de baja frecuencia en el receptor AS3933 de Austria Microsystems

El AS3933 ofrece menor consumo y mayor sensibilidad para aplicaciones portátiles de bajo coste.

Austriamicrosystems anunció el mejor receptor de 3 canales dentro de su clase, en un rango de frecuencia de 15 a 150 kHz con capacidad de ajuste automático de la antena. Este receptor permite la alimentación por bacteria, ya que añade la funcionalidad de despertador "wake-up". Además, con una sensibilidad de 80uVrms, tanto el rango de medida como el coste se optimizan mientras el mínimo consumo de 2.7uA aumenta de forma importante la disponibilidad de la batería y su vida útil.

PREMO, fabricante líder de componentes RFID, integra en el AS3933 su antena receptora 3DCoil de la serie 3DC15 (3DC15-1000J / 3DC15-0720), una antena de bajo perfil y elevada sensibilidad (la más alta del mercado) que ayuda a recudir coste, reducir espacio en la PCB e incrementar la fiabilidad del circuito. Además de la antena receptora, PREMO incluye la serie KGEA-BFCR (KGEA-BFCR-B-0500J), una antena emisora diseñada para permitir distancias de lectura largas. Ambas antenas son fabricadas en producción masiva en las plantas de producción que la compañía tiene en Tánger (Marruecos) y Wuxi (China).

PREMO ofrece al mercado un amplio rango de antenas emisoras (valores estándar de catálogo) de 20, 125 y 134,2kHz con diferentes combinaciones de valores de inductancia-condensador y potencia adecuada. Todos los códigos de la serie KGEA, (modelo BFCR, BFCWX, BFCAM, etc.), han sido diseñados con una ferrita especial (de bajas pérdidas) y condensador especial para aplicaciones de altos picos de voltaje. La antena cumple la normativa AEC-Q200, norma de calidad específica para automoción, y grado de protección IP68 con composición especial y caja de plástico de elevada resistencia. La versión estándar de la seria KGEA se entrega sin conector. En caso de requerimiento de conector éste se añadirá de acuerdo a la especificación del cliente (con o sin protección especial contra el ingreso de agua).

La serie 3Dcoil se ofrece en dos tamaños diferentes, 3DC11LP y 3DC15 y 3 formatos diferentes, etiqueta Standard, encapsulado, que ofrece mayor resistencia manteniendo prácticamente las mismas dimensiones (13x11.6x3.6 mm) y sin alterar sus parámetros eléctricos, y Foam, con sistema de absorción de impactos que mejora las características mecánicas del componente, especialmente en pruebas de caída (drop test). Estas antenas receptoras, cumplen la normativa AEC-Q200 y su aplicación principal es entrada pasiva sin llave al vehículo, sistemas de localización en tiempo real, control de acceso, etc.

Con una amplia gama de frecuencias - de 15 a 150 kHz - el AS3933 ofrece a los diseñadores una amplia gama de aplicaciones, incluyendo RFID activos, trazabilidad de artículos, sistemas de localización en tiempo real, identificación de personas, redes de sensores inalámbricos, control de acceso, o acceso remoto sin llave. El AS3933 proporciona un valor RSSI digital (indicador de intensidad de la señal recibida) para cada canal activo. También incluye un generador de reloj interno que se deriva de un oscilador de cristal de cuarzo o del oscilador RC interno, y es el primer dispositivo en proporcionar una antena integrada con sintonizador automático, que sintoniza la antena a la frecuencia deseada.

www.grupopremo.com

Brida medidora de par con certificación ATEX

Ideal para aplicaciones en gasoductos y entornos con peligro de explosiones.

Para responder a los requerimientos en tareas de análisis de rendimiento de una máquina, **HBM,** fabricante de equipos y componentes para la medida de magnitudes mecánicas y pesaje, cuenta con una gama completa de transductores de par que se integran directamente en la cadena cinemática, entre el accionamiento y la máquina.

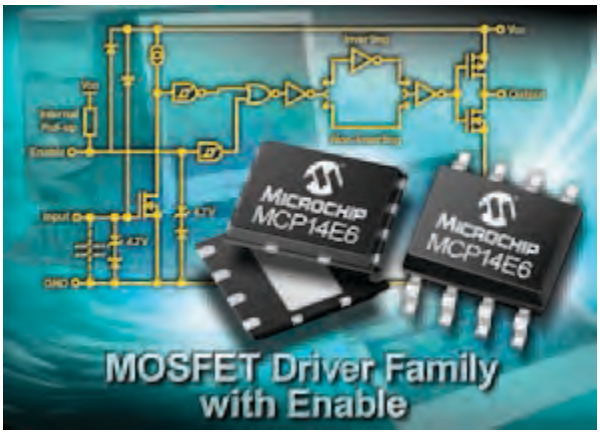
Del amplio abanico de modelos de HBM destacan las bridas medidoras de par T10FH que, disponibles con pares de giro nominales de serie de hasta 300 kN·m, incluyen este tipo de transductor y se pueden utilizar en entornos con peligro de explosiones. También existe una versión especial T10FH con la certificación ATEX ll 2G EEx d e q IIC T4 para rangos de medida nominales hasta 150 kNm y aplicación en la zona 1. La transmisión de datos entre el rotor y el estator es digital. De este modo, se consigue registrar y transmitir los valores medidos de forma correcta y segura, incluso bajo condiciones ambientales difíciles, tales como interferencias electromagnéticas o temperaturas oscilantes.

Una aplicación habitual del transductor de par T10FH-ATEX es el control de motores de gas para sistemas de compresor en gasoductos o cavernas subterráneas de almacenamiento de gas. Dado que la presión en el gasoducto oscila, se requiere una regulación rápida y precisa que reaccione a los parámetros variables de funcionamiento. De esta forma, se puede proteger eficazmente la aparición de cualquier daño, al tiempo que se reduce el consumo de combustible.

La alta precisión en la medición de par también permite registrar los datos de estado del sistema, proporcionando intervalos de mantenimiento óptimos en el marco del "seguimiento basado en la condición".

www.hbm.es

Microchip amplía su familia de controladores MOSFET



Microchip anuncia la ampliación de su familia de controladores MOSFET con los nuevos controladores MOSFET en el lado de bajo potencial (low-side) MCP14E6/7/8 y MCP14E9/10/11. Estos nuevos controladores complementan a los controladores en el lado de bajo potencial (low-side) de Microchip ya existentes de 4,5A MCP14E3/4/5 al ofrecer corrientes de pico de salida de 2A y 3A, respectivamente. Esta familia de bajo coste se caracteriza por un amplio rango de tensiones de trabajo de 4,5V a 18V e incorpora patillas de activación de entrada para ofrecer capacidad de apagado con el fin de ahorrar energía. Los controladores se suministran en encapsulados SOIC de 8 patillas y DFN de 8 patillas y 6mm x 5mm con el fin de reducir el espacio necesario en la placa para el control de fuentes de alimentación en aplicaciones de electrónica de consumo como servidores, ordenadores personales y ordenadores portátiles.

La ampliación de la familia de controladores MOSFET ayuda a los ingenieros a reducir el consumo de energía y a ofrecer más funciones en encapsulados más pequeños y con un bajo coste. Los nuevos dispositivos dobles MCP14E6/7/8 presentan una corriente nominal de salida de 2A, mientras que los controladores dobles MCP14E9/10/11 tienen una corriente de pico de salida de 3A. El amplio rango de tensiones de tensiones de trabajo de 4,5V a 18V ofrece soporte a una amplia variedad de tensiones de entrada y los pequeños encapsulados disminuyen los costes al reducir el espacio en la placa.

La incorporación de estos dispositivos a la oferta de productos analógicos y de interface de Microchip aumenta las opciones disponibles y la flexibilidad al proponer una línea completa de controladores de bajo coste en el lado de alto y de bajo potencial (high-side y low-side), con corrientes de pico a la salida entre 0,5A y 12A, suministrados en encapsulados de uso extendido.

Los controladores MOSFET MCP14E3/4/5, MCP14E6/7/8 y MCP14E9/10/11 están disponibles en encapsulados SOIC de 8 patillas o DFN de 8 patillas y 6mm x 5mm.

www.microchip.com

Convertidores DC-DC EcoSpeed™

Los reguladores POL SC173 y SC174 están optimizados para las aplicaciones "verdes" más demandadas.

Semtech Corp., empresa representada en España por **Anatronic, S.A.,** ha anunciado el lanzamiento de los reguladores buck SC173 y SC174, que suponen una nueva incorporación a la plataforma de convertidores DC-DC EcoSpeed™ con arquitectura avanzada para aplicaciones de punto de carga (POL) de próxima generación.

Diseñados para incrementar la eficiencia y responder a las necesidades de las aplicaciones "verdes" más demandadas, los nuevos dispositivos son ideales para automatización de oficinas, redes y equipos de comunicaciones; set-top boxes; productos portátiles; fuentes de alimentación; y otros muchos productos embebidos.

Los convertidores SC173 y SC174 se benefician del uso de la arquitectura EcoSpeed para lograr mejoras en modo de corriente convencional, modo de tensión y topologías de control on-time. Esta plataforma permite una respuesta ultra rápida ante transitorios y enorme eficiencia con cualquier carga, eliminando así la necesidad de componentes externos de compensación.

La arquitectura EcoSpeed, que favorece la creación de una solución de gestión eléctrica sencilla, compacta y económica, emplea un bucle de control avanzado para ajustar la sincronización basada en las tensiones de entrada y salida. Esto posibilita una operación de frecuencia pseudo-fija con ±15 por ciento de precisión y EMI previsible.

Los dispositivos SC173 y SC174 son programables de 200 kHz a 1 MHz para ayudar a los diseñadores a optimizar su fuente de alimentación y maximizar la eficiencia de conversión. Estos convertidores también incorporan tecnología SmartDrive™ de la representada de Anatronic para reducir EMI en aplicaciones sensibles al ruido.

La arquitectura EcoSpeed también permite trabajar con condensadores cerámicos para incrementar la flexibilidad y la fiabilidad de diseño. Esto, combinado con la ausencia de dispositivos externos de compensación, contribuye a reducir los costes y el tiempo de desarrollo.

El SC173 y el SC174 con corrientes de salida de 3 y 4 A, respectivamente, se encuentran disponibles en un encapsulado MLPD (3 x 3 x 1 mm) de 10-pad libre de halógenos.

www.anatronic.com

Fuentes de alimentación AC-DC de hasta 1kW para aplicaciones médicas e industriales

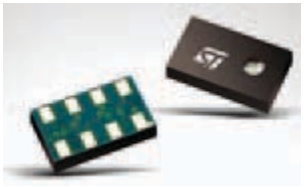
XP Power ha presentado las últimas incorporaciones a sus **series SHP y MHP** de fuentes de alimentación AC-DC de alta potencia con ventilador. Tanto la SHP1000 como la MHP1000 destacan por su elevada eficiencia, típicamente del 85%, tienen una densidad de potencia de 8.9 Watts/pulgada cúbica y son capaces de suministrar hasta 1000 W con un amplio rango de entrada de 90 a 264 VAC. Los modelos con salida de 24 VDC o más pueden suministrar hasta 1200 W si se dispone de más de 180 VAC a la entrada. El rango completo de 6 modelos ofrece salidas a +12, +15, +24, +28, + 36 o +48 VDC. La salida puede ser ajustada por el usuario en un +/- 10% de la nominal. Adicionalmente siempre disponen de una salida a +5VDC, 1A para propósitos de standby. La serie MHP1000 cumple las especificaciones de seguridad UL/IEC60601-1 para equipamiento médico, así como los límites EMC EN55022/EN55011 en cuanto a emisiones conducidas para las clase B e irradiadas para la clase A. La serie SHP1000 cumple con el estándar UL/IEC60950-1 para equipamiento ITE, junto con los requerimientos SEMI F47 y los límites EMC EN55022 del mismo modo que la serie MHP1000.

www.xppower.com

Sensor de presión con tecnología MEMS para incrementar la precisión en las mediciones de altura

El LPS001WP es ideal para teléfonos inteligentes y otros muchos dispositivos portátiles, estaciones meteorológicas y aplicaciones industriales.

STMicroelectronics, uno de los mayores fabricantes de sensores MEMS para aplicaciones de consumo y portátiles, ha presentado un nuevo sensor MEMS (Micro-Electro-Mechanical Systems) que permite establecer con exactitud las variaciones de altura de teléfonos inteligentes y otros dispositivos portátiles, desde 750 metros por debajo del nivel del mar a la cima del Everest.



El nuevo LPS001WP es un diminuto sensor de presión de silicio que se beneficia del uso de tecnología innovadora para ofrecer mediciones de presión con una resolución extremadamente alta y, por lo tanto, también de altitud, en un encapsulado HCLGA-8L ultra-compacto. Este novedoso modelo es ideal para teléfonos inteligentes, relojes deportivos y otros equipos portátiles, así como para estaciones meteorológicas y entornos industriales.

Una de las primeras aplicaciones esperadas es la mejora de dispositivos portátiles equipados con funciones GPS tradicionales que solo pueden identificar la localización del equipo en dos dimensiones. Con la incorporación del LPS001WP, dichos dispositivos podrán mostrar la ubicación exacta en las tres dimensiones, permitiendo que, por ejemplo, un teléfono móvil realice una llamada a los servicios de emergencia y sea localizado (incluso la planta del edificio en el que se encuentre).

El LPS001WP posee un rango operativo de presión de 300 a 1100 milibares, que corresponde a las presiones atmosféricas entre -750 y +9000 metros sobre el nivel del mar, y puede detectar cambios de presión de menos de 0.065 milibares, que equivalen a 80 centímetros de altitud.

El nuevo producto MEMS ha sido desarrollado utilizando una tecnología propia de ST, denominada “VENSENS”, que permite fabricar el sistema en un solo chip de silicio. De esta forma, se incrementa la fiabilidad y elimina la necesidad de una bobina ‘wafer-to-wafer’.

El sensor del LPS001WP se basa en una membrana flexible de silicio por encima de una cavidad de aire con un hueco determinado y presión definida. La membrana es muy pequeña en comparación con soluciones tradicionales ‘micro-machined’ de silicio y está protegida ante roturas mediante “tapones” mecánicos. La membrana incluye una piezo-resistencia, una estructura diminuta cuya resistencia eléctrica varía cuando la membrana se flexiona como respuesta a cambios en la presión externa. Esta variación en la resistencia es monitorizada, compensada térmicamente y convertida a un valor digital que se puede leer por el procesador host del equipo usando protocolos de comunicación I2C o SPI.

www.st.com

NUEVO

ELEKTOR OSPV¹

Proyecto de código libre sobre ruedas

ES PARA USO INTERIOR

Al final tú decides (y la legislación local) por dónde quieres conducir con este OSPV¹, pero la intención es que sea utilizado en el interior. No es apto para escalones, baches y suelos desiguales. Esto es porque el OSPV¹ tiene un margen respecto al suelo de tan sólo 2 centímetros.

Las dos ruedas, con un diámetro de 14 cm cada una, tienen una distancia suficiente como para mantenerte en equilibrio de forma estable, pero dejan demasiado poco espacio libre como para pasar fácilmente por un escalón; naves industriales, pasillos de los colegios, y por nuestra oficina - en cualquier sitio donde haya un suelo liso ahí va como la seda.

El conductor permanece levantado con ambos pies fuera de la ruedas. Hay que practicar un poco lo de montarse, pero el OSPV¹ da una sensación de estabilidad debido a que el punto de la gravedad está muy bajo.

SE CONDUCE FÁCILMENTE

Una vez que los pies estén colocados en el sitio correcto es cuestión de apoyarte hacia delante o hacia atrás para poner en marcha el OSPV¹. Si en el ElektorWheelie se tenía que mover la palanca de dirección hacia la izquierda o hacia la derecha para hacer los giros; en el OSPV¹ se hace con una pequeña palanca de mando. La reacción a la palanca de mando es instantánea - una pirueta y otros movimientos elegantes forman parte de sus posibilidades - ¡en un salón de baile no quedaría nada mal! Al conectarse se guarda la posición del OSPV¹ y se toma como referencia – ten cuidado de que las plataformas estén en horizontal durante este procedimiento.

ES LIGERO Y PLEGABLE

El OSPV¹ está diseñado de tal forma que puede ser desmontado fácilmente – así cabe en el maletero del coche. ¡Y es ligero! Con sus 25 kg cualquier hombre puede levantarlo sin ningún problema. La bisagra/punto de fijación está en la palanca de dirección. La palanca de dirección se separa en dos si se aflojan las dos tuercas con mariposas. Apostaría a que mucha gente haría aquí uso de su creatividad y encontraría una solución incluso más sencilla e ingeniosa. Todo eso está permitido – el OSPV¹ se presta muy bien a estas sencillas modificaciones.

El ElektorWheelie que lanzamos al mercado anteriormente, se hizo con un lugar entre escuelas y particulares que querían examinar a fondo la tecnología que hay detrás de los vehículos autobalanceados. Este Elektor OSPV1 está basado en la misma idea, sólo que con una diferencia: ¡es para uso interior, se conduce fácilmente, es ligero y plegable, hecho en código libre y además tiene un bonito aspecto!

Especificaciones del OSPV¹

Peso:	25 kg	Margen del suelo de las plataformas:	2 cm
Altura:	120cm	Altura de las plataformas:	5,6 cm
Anchura:	47cm	Anchura entre las plataformas:	29,5 cm
Profundidad:	47 cm	2x baterías:	batería de gel de plomo CTM ct9-12L, 9 Ah 12V
Carga máxima:	90 kg		

¿Quién no está buscando equilibrio en su vida? Encontrarlo no es tan complicado con algo de electrónica moderna. El año pasado lanzamos el ElektorWheelie - un vehículo autobalanceado. En esta edición te damos un paseo con el OSPV1 - El Open Source People Vehicle 1.

ES CÓDIGO LIBRE

Esto es lo más divertido del OSPV1 – se puede cambiar todo a gusto del usuario. Su electrónica es idéntica a la del ElektorWheelie, la programación está disponible y se pueden hacer modificaciones para cambiar el comportamiento de conducción. Todo aquel que haya desarrollado una ampliación para el ElektorWheelie puede utilizarla en el OSPV1. Los esquemas, el diseño de la placa y el listado del código fuente están disponibles en el sitio Web de Elektor [1]. El OSPV1 se suministra con cargador para las dos células de batería de 12 V con una capacidad de 9 Ah.

Y TIENE UN BONITO ASPECTO

Lo sé — los gustos son diferentes, pero hasta hoy el OSPV1 sólo ha recibido reacciones positivas. Oyes cosas como “ahhh”, “ohhh”, “gracioso”, “cosa divertida” y sigue así. La construcción está hecha en código libre, se ha mantenido ligera y tiene un acabado impecable.

Y... QUÉ ES LO QUE PUEDO HACER CON ÉL

En primera instancia el OSPV1 está pensado para el desplazamiento de personas, pero... no hace falta que siga siendo así. Podrías inventar otras aplicaciones que varían desde una carretilla eléctrica hasta una útil ayuda para las compras. Aquí se hace notar la ventaja del código abierto. Todo el mundo es libre de utilizar este diseño y construir algo especial. Como hemos dicho anteriormente es para ser utilizado en el interior. Puedes llevarlo a un ámbito más amplio: naves industriales, terminales, pasillos de colegios, mientras el suelo sea liso, el OSPV1 seguirá circulando.

Y en la calle... quizás lo repetimos demasiado, pero queremos mencionar que hay normas (por países) para la conducción de este tipo de medios de transporte. También hay un límite en peso del conductor – hasta 85 kg todo va perfecto. Además, todo el mundo debería pensar en la protección de su cuerpo, ¡sigue siendo un medio de transporte!

DONDE SE PUEDE ADQUIRIR

El OSPV1 se vende exclusivamente en la tienda Web de Elektor. Puedes encontrar toda la información en [1].

Radio de acción:	8 km	Motoren:	DC 2x250 W
Diámetro de giro:	0 m	Transmisión:	correa dentada HDT
Velocidad máxima:	15km/h	Adaptador de carga:	apto para EEUU y EU
Ruedas:	PU, 14 cm en diámetro	Tiempo de carga:	2,5 horas

Curso de audio DSP

Parte 2: programación del DSP

El DSP6375 de Freescale se programa mejor usando su propio lenguaje ensamblador. Aunque esto pueda parecer anticuado -un resto no deseable de los primeros días de los ordenadores digitales- no ocurre esto con el DSP56374. Esto se debe, en parte, a que las transacciones típicas de datos en paralelo de la aritmética del tratamiento de señales no pueden ser manejadas fácilmente en un lenguaje de alto nivel, y en parte porque el código generado usando un lenguaje de este tipo no está tan bien optimizado, en lo que a nivel de ejecución se refiere, como el código generado usando el lenguaje ensamblador, principalmente porque la mayoría de los programas DSP tienen estructuras muy simples. Aquí, las principales barreras en el proceso de programación provienen de los rigurosos requisitos de precisión, donde incluso los lenguajes de alto nivel no ofrecen ninguna ayuda al respecto.

Alexander Potchinkov (Alemania)

El lenguaje ensamblador de la familia DPS563xx es relativamente fácil de aprender. Tiene una alta consistencia estructural y está basado en un modelo de programación bien concebido, que también se beneficia de que los fabricantes pueden aprender de los DSPs existentes, antes de lanzar sus propios productos, y sobre todo, de lo que ellos pueden aprender de la experiencia con sus propios procesadores.
Necesitaremos tres programas basados en PC para el desarrollo de aplicaciones: un ensamblador, un simulador y un depurador (debugger). Estos programas están disponibles como programas individuales o como herramientas de aplicaciones de un paquete, llamado Entorno de Desarrollo Integrado (IDE). Lo que necesitamos esencialmente es un ensamblador, pero trabajar sin simulador ni debugger supone hacer el trabajo sin las herramientas más importantes para encontrar y corregir errores en los programas del DSP. El ensamblador convierte el código fuente en el código objeto, mientras que el debugger y el simulador nos ayudan a localizar y corregir errores en el programa. Estas dos últimas herramientas son muy similares y tienen casi las mismas funciones y rasgos, con la diferencia básica de que un debugger necesita tener un DSP real conectado, mientras que un simulador no.

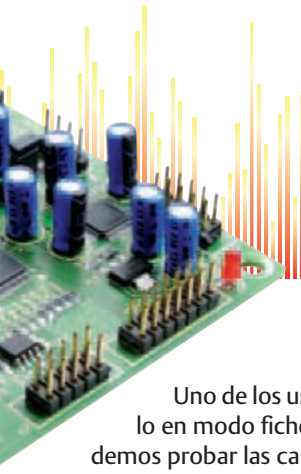
Ensamblador

El punto de partida es un fichero de texto que contiene un programa DSP escrito en lenguaje ensamblador. Este archivo de texto puede ser generado usando cualquier editor que queramos. Cada instrucción puede tener hasta seis campos independientes, separados por espacios: Un comentario comienza con un punto y coma. Del archivo de código fuente, el ensamblador genera un archivo que contiene el código objeto. Este archivo tiene la extensión `.cld`. Podemos usar este archivo con el simulador o podemos usar el depurador para cargarlo en el DSP y ejecutarlo en el DSP.

Simulador

El simulador es una herramienta muy útil para el desarrollo de aplicaciones. Hablando llanamente, el simulador es una emulación de un DSP que se ejecuta en el entorno de un PC, con velocidad de ejecución reducida y que carece de los dispositivos periféricos del DSP. La emulación incluye no sólo el camino de datos del DSP, sino también todos los puertos y la memoria completa, así como el sistema de interrupciones. Como el simulador también emula el pipelining (o segmentación), esto nos permite poder contar el número de ciclos de reloj del procesador DSP necesarios para ejecutar cualquier rutina de programa deseada, algo que es, por supuesto, una información importante para el tratamiento en tiempo real. La obtención de esta información directamente del programa real del DSP sería una tarea muy laboriosa. El simulador también puede ser usado para el debugging, ya que los registros virtuales y las posiciones de memoria del DSP pueden ser leídos y escritos en cada paso de un programa. Por ejemplo, podemos identificar que el comportamiento incorrecto del programa se debe a un valor erróneo en un registro, o podemos cambiar el valor en un registro en un punto particular del programa y permitir que el programa siga ejecutándose desde ese punto, para poder ver cómo se comporta con el nuevo valor cambiado.
El simulador es ejecutado con el código objeto. Esto equivale a la carga del programa en el DSP real, en el sentido de que el código del programa es cargado en la imagen de la memoria del DSP del simulador. Podemos comprobar el funcionamiento del programa en modo “paso a paso” o en modo “bloque”. Para ello, el simulador ofrece puntos de ruptura condicionales e incondicionales, contadores de ciclos de reloj del procesador y de las instrucciones, un ensamblador de una sola línea, y la opción de mostrar toda el contenido de la memoria y los registros, bien individualmente o bien por bloques, y guardarlos como archivos en formato ASCII.

Etiqueta	Operador	Operandos	Bus de Transf. X	Bus de Transf. Y	Comentario
`Etiqueta`	`mac`	`x0,y0,a`	`x:(R0)+,x0`	`y:(r4)+,y1`	`; operación MAC`



Uno de los usos principales del simulador es el de ejecutarlo en modo fichero de E/S (Entrada-Salida). En este modo podemos probar las características del tratamiento de señales de un programa DSP, usando datos de prueba ubicados en un archivo (la señal de prueba), como entrada, y escribiendo los datos de salida del programa (la señal procesada) en otro fichero, el cual puede ser analizado usando un programa estándar.
Para ilustrar como usar el simulador, presentamos aquí un sencillo ejemplo basado en el fichero fuente `fileio.asm`, que el ensamblador convierte en el archivo objeto `fileio.cld`.

```
    org x:$0 ;
simdatain  ds 1    ; asignación de memoria para el dato de entrada
simdataout ds 1    ; asignación de memoria para el dato de salida
    org p:$100
start  move x:simdatain,x0  ; x0 <- dato de entrada
    move x0,x:simdataout    ; x0 -> dato de salida
    jmp start               ; repite el proceso
```

En el programa DSP, el contenido de la posición de memoria `simdatain` del DSP es copiado a la posición de memoria `simdataout` del DSP, usando el registro x0 para el almacenamiento temporal. Este proceso se repite indefinidamente, algo que se consigue por la inclusión de la instrucción de salto `jmp start`. Se genera un fichero de datos llamado `Infile.dat` al ejecutar el programa en modo simulador. Los comandos del simulador mostrados más abajo leen los datos del archivo de entrada y generan un fichero de salida llamado `outfile.dat`, el cual, en este caso, debe contener los mismos datos que el archivo de entrada.
Las funciones de las instrucciones individuales del simulador son:

```
(1)   reset s ; reinicia el simulator
(2)   input off ; reinicia todos los ficheros de entrada
(3)   output off ; reinicia todos los ficheros de salida
(4)   load fileio.cld ; carga el programa DSP
(5)   input #1 simdatain infile.dat -rh ; entrada desde infile.dat
(6)   output #1 simdataout outfile.dat -rh -o
            ; salida hacia outfile.dat
(7)   break EOF ; para cuando un fichero de entrada alcanza su
            "end-of-file"
(8)   go ¡; ejecuta el programa
```

- Las instrucciones de las líneas 1 a 3 reinician el simulador y cierran cualquier archivo de entrada y salida actualmente abierto.
- La línea 4 carga el programa DSP.
- La línea 5 carga el archivo de entrada. La opción `-rh` indica que los datos de entrada deben ser leídos como números hexadecimales. La opción alternativa `-rf` puede ser usada para leer números fraccionales, los cuales son importantes para el tratamiento de señales.
- La línea 6 abre el archivo de entrada. La opción `-o` especifica que el archivo existente con el nombre dado en la instrucción ha de ser sobrescrito.
- La línea 7 dice al simulador que el proceso debe ser detenido cuando se haya alcanzado el final de los datos de entrada.
- Finalmente, la línea 8 inicia la ejecución del programa DSP en modo simulador.

Las instrucciones del simulador pueden ser salvadas en un archivo de comandos.

Debugger

El debugger, en esencia, hace lo mismo que el simulador, pero con el programa ejecutándose en el DSP real en lugar de en el DSP simulado. Podemos usar el debugger para cargar programas en el DSP y ejecutarlos en modo paso a paso o con puntos de ruptura. El debugger puede colocar al DSP en un modo especial de funcionamiento en el que todos los contenidos de los registros y de la memoria pueden ser leídos o escritos. La velocidad de ejecución es mucho más rápida que con el simulador, ya que el programa corre en el circuito real.
En vez de usar el método del fichero de entrada-salida para la prueba y los ficheros de la señal de respuesta, podemos probar nuestros programas DSP a través de los interfaces de audio, por ejemplo, usando a un editor de onda para generar las señales de prueba.

Utilización del software

La **figura 1** muestra la relación entre los programas que se ejecutan en el ordenador (cuadros rectangulares con sombreado gris) y los archivos necesarios y/o generados en este contexto (mostrados en cajas con esquinas redondeadas). El punto de partida es el código fuente, que es convertido en el código objeto por el ensamblador. Cuando ejecutamos el ensamblador, podemos usar la opción `-l` para especificar la creación de un archivo lista que, a menudo, es muy práctico. En nuestro caso, necesitamos obligatoriamente el código objeto que podemos generar especificando la opción `-a` al ensamblador. El código objeto puede ser introducido en tres programas diferentes: el debugger (con un DSP conectado al PC), el simulador, o en un programa llamado *srec*, que aún no hemos mencionado. Podemos usar este programa para generar archivos S, los cuales son necesarios para programar un "bootstrap" (proceso por el que se carga el programa de trabajo, o SO, en memoria) en la memoria PROM. Los dispositivos de programación comúnmente usados pueden trabajar con estos archivos S.

Entornos de Desarrollo de Programas

Necesitamos dos cosas para los programas de desarrollo del DSP: las herramientas de los programas de desarrollo y un adaptador para conectar el programa depurador, que corre en el PC, al DSP.

Suite 56 de Freescale

Software

El programa, formado por un ensamblador, un "linker", un simulador y un debugger, lo podemos descargar gratuitamente de la página web de Freescale [1]. El simulador y el debugger están disponibles en dos formas: con un interfaz de usuario en la línea de comandos o con un interfaz de usuario gráfico (GUI56300 para el simulador y GDS56300 para el debugger). El programa tiene una curva de aprendizaje relativamente corta. La **figura 2** muestra los componentes del programa de la Suite 56. La Suite 56 está pensa-

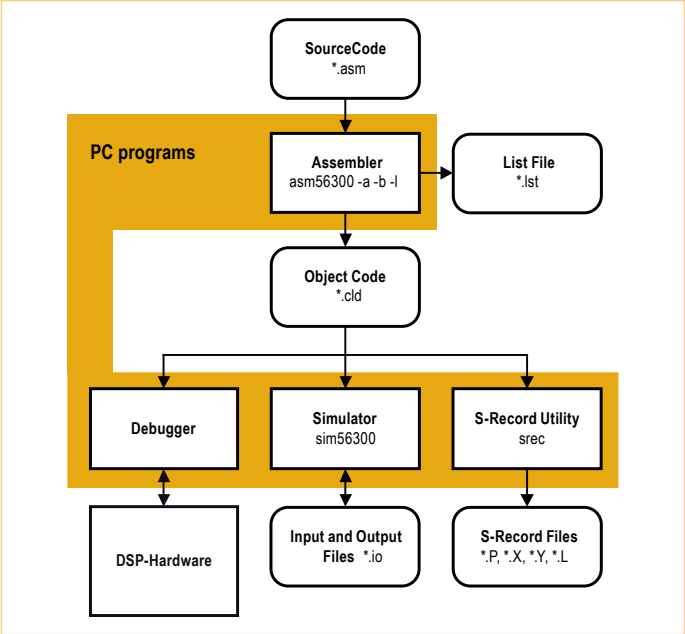


Figura 1. Relaciones entre programas y ficheros en el entorno de desarrollo de Freescale.

da, sobre todo, para el uso en un ambiente de equipo, gracias a la presencia de un "linker" y varias librerías.

Adaptador

El enlace con el DSP se realiza a través del puerto paralelo del PC y de un adaptador, que podemos obtener de Freescale [2] o (de forma idéntica) de Macraigor, bajo el nombre de "Wiggler" [3]. También podemos construir el adaptador por nosotros mismos ya que todo lo que necesitamos es algún tipo de controlador de bus y unos pocos componentes. Los diseños detallados y la documentación para ello los podemos encontrar en la web, como por ejemplo el UniDSP56 de Gerrit Buhe [4]. Si nuestro PC carece de puerto paralelo, podemos montar una tarjeta PCI con un puerto paralelo.

Symphony Studio de Freescale

Software

Freescale también dispone de un ensamblador, "linker", compilador C, simulador y debugger como "plugins" gratuitos para el Entorno de Desarrollo Eclipse [5]. Sin embargo, la configuración de estos "plugins" para crear un programa en el entorno de desarrollo no está bien documentada y, si no estamos familiarizados con Eclipse, nos puede causar algunos problemas. Por otra parte, sí que vale la pena aprender cómo usar entornos de desarrollo de aplicaciones modernos, ya que pueden ser usados con muchos tipos de procesadores diferentes.

Adaptador

Al igual que Estudio 56, el enlace con el DSP se proporciona por el puerto paralelo del PC, con la ayuda de un adaptador, que podemos obtener de Freescale [2] o (de igual forma) de Macraigor, bajo el nombre "Wiggler" [3]. Otra opción es usar un adaptador USB, como el usbWiggler de Macraigor [6] o el USB-EMU de Domain Technologies [7].

BoxView de Domain Technologies

Software

Domain Technologies produce una herramienta de depuración software que se vende bajo el nombre de BoxView [8]. Podemos usar esta herramienta en combinación con las herramientas ensamblador y simulador gratuitas de Freescale (asm56300 y sim56300).

Adaptador

Está disponible un adaptador USB [7] de Domain Technologies.

Paquete de depuración EVM de Domain Technologies

La marca Domain Technologies dispone de una herramienta de depuración, la BoxView (Bv30xEvm.exe), para ser usada con la tarjeta del módulo de evaluación de Freescale (EVM) para los procesadores digitales de señal DSP563xx, que podemos descargar de forma gratuita [9], en caso de que no dispongamos de una copia de paquete EVM. Esta herramienta es, ante todo, fácil de usar y es el debugger favorito del autor. La **figura 3** muestra un pantallazo de un ejemplo de una sesión de depuración usando la herramienta BoxView. Junto con un panel que muestra el código fuente, podemos ver otros paneles que muestran los contenidos de los registros y la memoria. El contenido de la memoria también se puede mostrar de forma gráfica, que a menudo puede ser muy útil.

Adaptador

Una opción es conseguir una placa EVM (a veces, disponible de segunda mano a precio de amigo) y usarla como un adaptador para una placa DSP independiente.

Junto con los adaptadores habituales (disponibles comercialmente) y con los adaptadores DIY que se conectan al puerto paralelo de un PC, el autor y Elektor colaboran actualmente en el desarrollo de dos adaptadores USB baratos pensados para ayudar a los lectores de Elektor a alcanzar la velocidad en el mundo del tratamiento digital de las señales de audio, en combinación con el hardware para este curso DSP.

Código DSP y bucle de audio

Todos los programas DSP de esta serie de artículos tienen la misma estructura de alto nivel, que consiste en una estructura de programa y un bucle de audio.

La estructura de programa es, en gran parte, la misma para todas las aplicaciones: contiene el código de configuración y el código para el funcionamiento de los interfaces periféricos DSP. Además de esto, realiza la inicialización del SRC y de la aplicación, así como el borrado del estado de las memorias, para asegurar que el programa siempre comienza en las mismas condiciones. El bucle (o lazo) de audio contiene el código para la sincronización y el tratamiento digital de la señal.

En la estructura del programa también se configuran dos juegos de canales de entrada y salida. Están disponibles en la forma de un receptor de dos canales, en el puerto de datos de audio RX0, y en un transmisor de dos canales, en el puerto de datos de audio TX0. El programa DSP necesita dos "buffers" de memoria, que son usados como "buffers" de datos de audio. Dichos "buffers" están configurados en la memoria X RAM y equipados con punteros a las direcciones base de éstos:

```
TxBuffBase   EQU   $000000   ; Dirección Base del TX-Buffer (X)
RxBuffBase   EQU   $000010   ; Dirección Base del RX-Buffer (X)
TXPTR        EQU   $000020   ; Dirección del Tx-BufferPointer (X)
RXPTR        EQU   $000021   ; Dirección del Rx-BufferPointer (X)
```

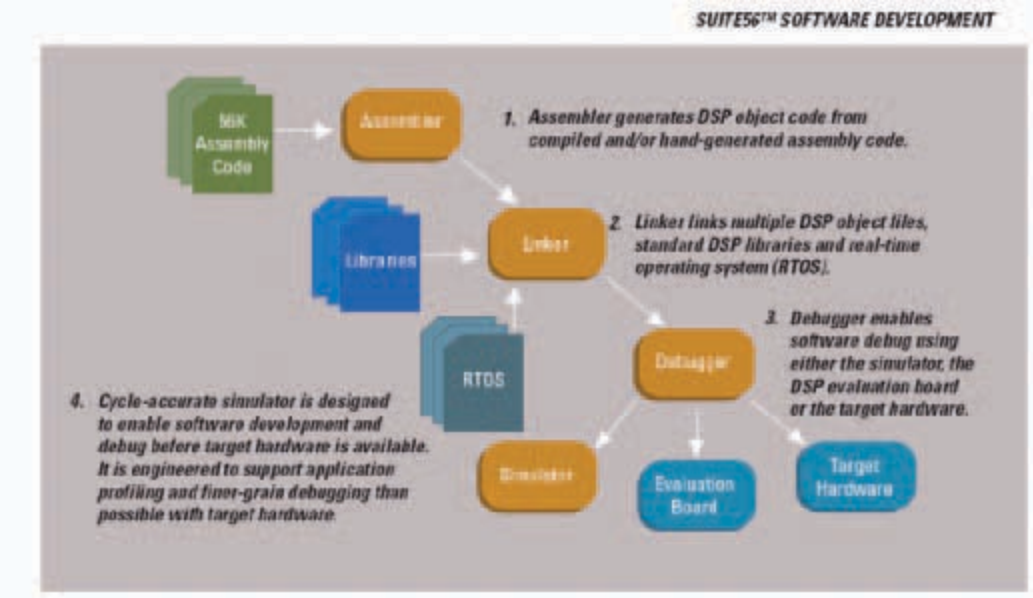


Figura 2. Herramientas de la aplicación Suite 56 (imagen cortesía de Freescale).

También necesitamos una bandera llamada *left/right flag* (`LRFlag` o *bandera izquierda/derecha*), que es escrutada en el lazo principal del programa para poder realizar la sincronización de los datos de audio.

```
LRFlag      EQU   $000022  ; Dirección de la Left/Right-Flag
RightRx     EQU   0        ; Bit de Posición de la Left/Right-Flag
```

Con una transmisión de datos de audio de dos canales, la muestra del canal izquierdo es transmitida en la primera ranura ("slot"), y la muestra de canal derecho se transmite en la segunda ranura, mientras que el DSP cuenta estas ranuras como 0 y 1. Para este propósito, los interfaces de audio del DSP trabajan en "modo red", el cual puede ser configurado para dos canales. En "modo red" el sistema de interrupción de los interfaces de audio genera una interrupción cuando la última ranura ha sido leída. El interfaz puede controlar hasta 32 canales de audio para cada puerto de datos de audio, pero aquí sólo necesitamos dos de estos canales. Esto significa que la última ranura alberga los datos del segundo canal, que es el canal derecho. La bandera izquierda/derecha, que tiene un valor lógico 1 cuando se han recibido los datos del canal derecho, es configurada en una rutina del servicio de interrupción (ISR), disparada por la interrupción "*Last Slot*" (es decir, *Última Ranura*).

En la sección de inicialización, los dos "buffers" de audio están llenos de ceros y se reinicializa la bandera de sincronización:

```
org    x:TxBuffBase
       dc 0
       dc 0
       endm
   org    x:RxBuffBase
       dc 0
       dc 0
endm
   org    x:LRFlag
       dc 0
```

El programa usa seis interrupciones largas ISRs para hacer funcionar los interfaces de audio. Las instrucciones `jsr` correspondientes y las direcciones base ISR son introducidas en la tabla del vector de interrupción.
Ahora ya estamos preparados para ejecutar el código DSP. La primera tarea es configurar los interfaces de audio. Las configuraciones para estos deben hacerse de forma individual, ya que dependen

Figura 3. Ventana del debugger BoxView de Domain Technologies.

de los componentes que sean usados. La configuración más importante determina si el DSP actúa como master o slave de audio. Si el DSP es master de audio, los relojes de los datos deben obtenerse del reloj del procesador DSP. En esta situación es aconsejable hacer que la frecuencia de reloj del oscilador sea un número submúltiplo entero de la frecuencia de reloj del master de audio. Por ejemplo, si la frecuencia de reloj del master es 24,576 MHz podemos usar un cristal de 6,144 MHz estándar y configurar el reloj del PLL del DSP con un factor de multiplicación de 24, proporcionando una frecuencia de reloj al procesador DSP de 147,456 MHz. Ahora, las señales de reloj del interfaz de audio pueden ser sacadas dividiendo la señal de reloj del procesador por 6, para obtener la señal de reloj maestra, y por 3072, para obtener la señal de la velocidad de muestreo (48 kHz), la cual sincroniza los datos de los canales izquierdo/derecho. Si el DSP debe trabajar como slave de audio, las señales de reloj requeridas pueden ser generadas por un conversor A/D o un receptor digital de audio que actúe como master. Para poder configurar los interfaces de audio, también tenemos que especificar el modo red, el receptor y transmisor a ser usados, el formato de datos, las interrupciones que se utilizarán, etc.
El lazo de audio comienza buscando la bandera sincronización:

```
AudioLoop
    jclr  #RightRx,X:LRFlag,* ; ¿palabra derecha recibida?
    bclr  #RightRx,X:LRFlag   ; reinicio bandera de sincronización
```

Aquí la instrucción `jclr` realiza la búsqueda. El lazo de búsqueda se ejecuta repetidamente hasta que la bandera sea distinta a cero. Se consigue una búsqueda repetida de la bandera usando un asterisco (`*`) como destino del salto. La subsecuente instrucción `bclr` reinicia la bandera a cero para que pueda ser configurada otra vez por la ISR correspondiente.
El siguiente segmento de código escribe primero las dos palabras del "buffer" de recepción en los dos registros `a` y `b` del acumulador .

```
        move    x:RxBuffBase, a   ; Canal Izquierdo -> a
        move    x:RxBuffBase+1, b ; Canal Derecho -> b

; inserta aquí el tratamiento de la señal

        move    a, x:TxBuffBase   ; a-> Canal Izquierdo
        move    b, x:TxBuffBase+1 ; b-> Canal Derecho
        jmp     AudioLoop
```

Después de esto, el dato de audio puede ser procesado por el código del DSP. Al final de este tratamiento, los seis valores de señal generados por el código de DSP son escritos al "buffer" de transmisión. En el código ejemplo, los dos registros acumuladores `a` y `b` son leídos para este fin. En ausencia de cualquier tratamiento de la señal, el DSP pasa los datos recibidos y un reloj de muestreo a la salida, sin ningún retardo. La instrucción final salta atrás, al lazo de búsqueda de la bandera sincronización.

En la siguiente entrega...

Esto completa nuestra introducción a la programación de DSPs. En la tercera entrega de esta serie describiremos el hardware desarrollado expresamente para este curso.

(110002)

Enlaces en Internet

[1] **Aplicación Suite 56 de Freescale:** www.freescale.com/webapp/sps/site/prod_summary.jsp?code=CW-SUITE56&fsrch=1&sr=4

[2] **Adaptador Freescale:** www.freescale.com/webapp/sps/site/prod_summary.jsp?code=DSPCOMMPARALLEL

[3] **Macraigor Wiggler:** www.macraigor.com/wiggler.htm

[4] **Diseño del adaptador DIY UniDSP56 de Gerrit Buhe:** www.unidsp56.de/downloads.html

[5] **Aplicación Symphony Studio de Freescale:** www.freescale.com/webapp/sps/site/prod_summary.jsp?code=SYMPH_STUDIO&fsrch=1&sr=16

[6] **Macraigor usbWiggler:** www.macraigor.com/usbWiggler.htm

[7] **Adaptador de Domain Technologies:** www.domaintec.com/usbemu.html

[8] **Herramienta de debugging BoxView de Domain Technologies:** www.domaintec.com/BoxView.html

[9] **Paquete de Herramientas de debugging EVM de Domain Technologies:** www.domaintec.com/FSsoftware.html

Sobre el autor

Alexander Potchinkov es catedrático del área de tratamiento digital de señales en la Universidad Técnica de Kaiserslautern y dirige una empresa de ingeniería para el tratamiento de señales de audio. Junto a los DSPs y sus algoritmos también ocupa su tiempo con los amplificadores de válvulas y la simulación en SPICE.

El Controlador ARM de Stellaris pasa a ser Biológico

Consideraciones de diseño para una incubadora de huevos

Tianyu Chen (de China)

Este circuito microcontrolador fue desarrollado en respuesta a desafíos planteados por un proceso biológico. Aunque la aplicación final, un sistema fiable para incubar huevos de pollo, pueda parecer especializada, el método de llegar hasta aquí tiene el potencial para otros muchos sistemas donde los parámetros como temperatura, movimiento y humedad, tienen que ser supervisados y controlados “con la diligencia de una gallina”.

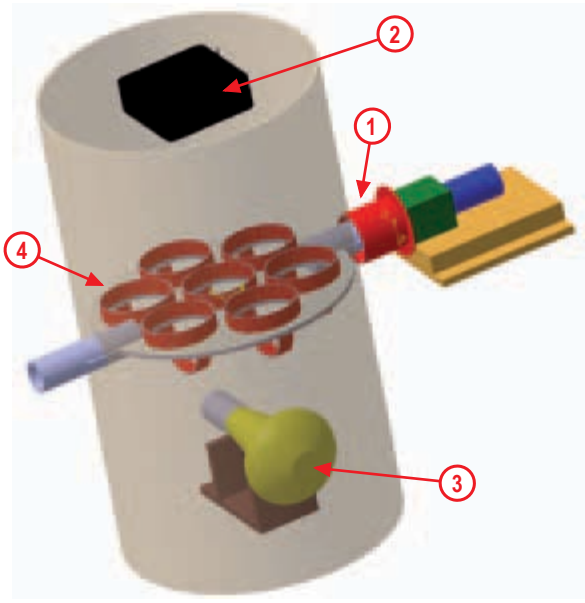


Figura 1. Impresión del artista de la cámara de incubación para huevos. 1: control de las marchas y eje del motor. 2: ventilador de PC. 3: bombilla de 25 vatios. 4: bandeja para los huevos.

Imitar una gallina doméstica incubando sus huevos, es decir, desarrollar una incubadora (máquina para incubar huevos), está lejos de ser un trabajo trivial. La tarea requiere habilidades y conocimientos tanto mecánicos como electrónicos, en combinación con algo de investigación en biología. Mejor que presentar un producto con la ingeniería ya hecha, vamos a cambiar el objetivo de este artículo que, así, será el de mostrar el proceso de varios factores como: limitaciones del mundo real, electrónica, software y mecánica, analizando y combinando todos ellos.

Los factores de mundo real

Podríamos pensar que la temperatura es el factor más importante durante el proceso de incubación de un huevo. Sin embargo, la humedad y otros factores también son significativos. Basándose en la investigación y literatura sobre estudios publicados, el autor fue capaz de poner en una lista los cinco puntos principales que deben ser observados, en este orden de importancia: 1. temperatura; 2. humedad relativa; 3. rotación; 4. ventilación; 5. higiene.

Temperatura. Como el embrión no tiene ninguna capacidad de autorregulación de temperatura, se requiere un dispositivo de control de temperaturas externo para que se desarrolle normalmente. Aunque algunos huevos se incuben con éxito a una temperatura de entre 35 y 40,5 grados centígrados, la temperatura óptima para los embriones es de 37,8 °C. Si bien a una temperatura más

Nota. Los Proyectos de los lectores son reproducidos basados en la información suministrada por el autor (es) sólo.
El uso de esquemas eléctricos y otras ilustraciones con el estilo de Elektor en este artículo o la disponibilidad de descargas de los programas del proyecto desde la página web de Elektor, no implica que el proyecto que haya pasado por los Laboratorios de Elektor para hacer una réplica que verifique el funcionamiento que se afirmada.

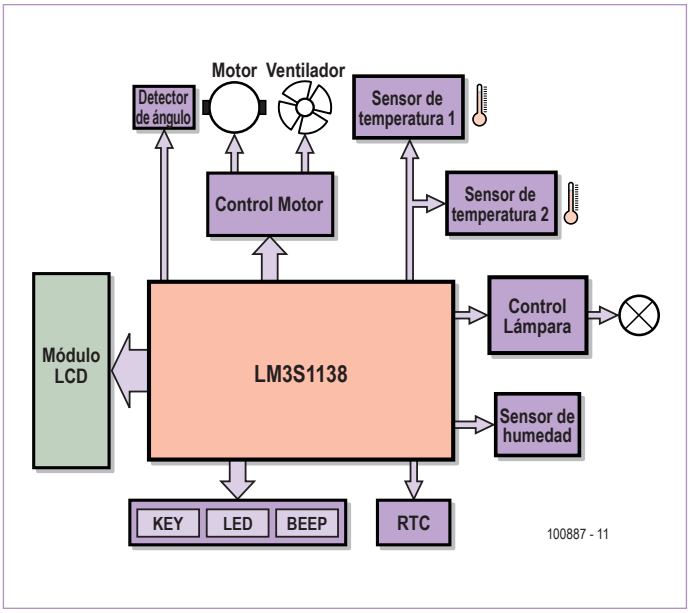


Figura 2. Elementos que forman parte en la fabricación del circuito de control, basado en un microcontrolador ARM, LM3S1138, M3 de Cortex.

elevada se acelerará el desarrollo embrionario, también se aumentará el índice de mortalidad y la calidad del pollo caerá. Por lo tanto, la temperatura debe ser lo más constante y aproximada posible a 37,8 °C.

Humedad relativa. Normalmente, la humedad relativa durante el desarrollo del embrión estará en el rango del 40-70 %, pero los valores óptimos están entre el 50 % y el 60 %. Si se mantiene la humedad apropiada, el huevo absorberá calor regularmente, incluso desde los primeros días de incubación, y disipará calor fácilmente cerca del final del período de incubación.

Rotación. Los huevos deben ser girados para prevenir la adherencia de embrión y estimular el movimiento de la bolsa amniótica. Idealmente el huevo debería ser girado de 6 a 8 veces al día, sobre todo durante las dos primeras semanas.

Ventilación. Durante el desarrollo embrionario en la cámara de huevo, el intercambio de aire siempre está en progreso, excepto durante unos pocos días, al principio del proceso. A medida que el embrión crece, la necesidad de oxígeno aumenta. Idealmente el contenido de oxígeno debería ser del 21 %, mientras que los niveles de dióxido de carbono deberían permanecer por debajo del 0,5 %. Valores de CO_2 que superen el 1 % harán que muchos embriones mueran.

Higiene. Aunque el embrión está rodeado por una sustancia coloidal y protegido por una cáscara y una cutícula, algunos gérmenes consiguen llegar hasta él, reduciendo las posibilidades de una incubación exitosa. Por lo tanto, se debe poner gran atención a la higiene y desinfección del sistema de incubación.

Una incubadora casera (DIY o Hágalo Usted Mismo)

Se dedicó un gran esfuerzo en la búsqueda de un dispositivo que pudiera ser convertido en una incubadora de huevos. La cámara incubadora debe tener unas buenas propiedades de aislamiento, ventilación adecuada y espacio suficiente para siete huevos. Un viejo termo eléctrico de Sanyo, que encontramos en la cocina, resultó ser el billete para el éxito.

En la **Figura 1** se intenta mostrar una representación artística de cómo se ha conseguido unir la funcionalidad del hardware a los factores tratados anteriormente. En la parte alta tenemos un ventilador (2), recuperado de un viejo ordenador (PC). En el centro encontramos una bandeja (4) que puede alojar los huevos durante el período de incubación. La bandeja es contralada por un motor engranado coaxial (1) con el fin de girar los huevos a intervalos de tiempo.

Las condiciones de aire en la cámara son todas importantes – es necesario disponer del suficiente oxígeno, pero la temperatura del aire también debe ser estable. Si entra demasiado aire frio en la cámara a un ritmo rápido, la temperatura del aire corre el peligro de bajar demasiado rápido. Encontramos un dispositivo de equilibrio térmico con la forma de una bombilla estándar de 25 vatios (3), que actúa como fuente de calor. Tan pronto como el aire caliente alcanza la parte superior de la cámara, el ventilador lo moverá hacia abajo y, al mismo tiempo, también añadirá un poco aire fresco por el agujero de la tapa. Con aire fresco y aire caliente mezclado a una buena temperatura, el aire dentro de la cámara tendrá una temperatura constante y una proporción de oxígeno alta.

La electrónica

Después de las consideraciones biológicas y mecánicas viene la electrónica, cuya función es la de conseguir que todo esto ocurra. En el corazón del circuito de control desarrollado se sitúa un microcontrolador ARM LM3S1138 de Cortex™, basado en el microcontrolador M3 [1] de Lumbrera Micro (ahora propiedad de Texas Instruments). Esta CPU pertenece a la serie "Stellaris®".

La diagrama de bloques de la **Figura 2** muestra los periféricos alrededor de la CPU como el teclado, el LED, y el 'Bip', que permiten que el usuario establezca las características de incubación y mostrar su estado. Cuando se enciende el sistema, éste pide al usuario indicar el tipo de huevo y el tiempo de incubación deseado. Durante el proceso de incubación la pantalla LCD muestra la temperatura y humedad reales en la cámara y 'los días que quedan para eclosionar'. Si la temperatura o la humedad son más altas o más bajas que el nivel de seguridad, o el sistema ha detectado una condición excepcional, los "bips" comienzan a sonar, alertándonos

para que tomemos medidas y evitemos la muerte de embrión.
El sistema tiene dos sensores de temperatura y un sensor de humedad en el interior. Los dos sensores de temperatura están localizados de tal modo que permiten detectar si la ventilación es pobre y es eliminada por la acción del ventilador. Además, si un sensor de temperatura está dañado o con un funcionamiento defectuoso, el otro actúa como sensor de reserva.
El circuito actual se extiende sobre dos placas de circuito impreso. El placa principal contiene la CPU, el interfaz de depuración, el circuito de reinicio, el teclado y el LED. La otra placa contiene los periféricos como el RTC y el controlador de motor. Debido a las limitaciones de espacio, no se ha incluido en este artículo el esquema eléctrico completo del sistema - sin embargo, sí está disponible para su descarga gratuita en la página web de Elektor [2]. Para los componentes principales tenemos: CPU = LM3S1138; controlador de motor = L298N; RTC = DS1305; sensor de humedad = HS1101/NE555; sensores de temperatura = LM75. Siendo imparciales, nada demasiado esotérico aparte de la CPU de Luminary.

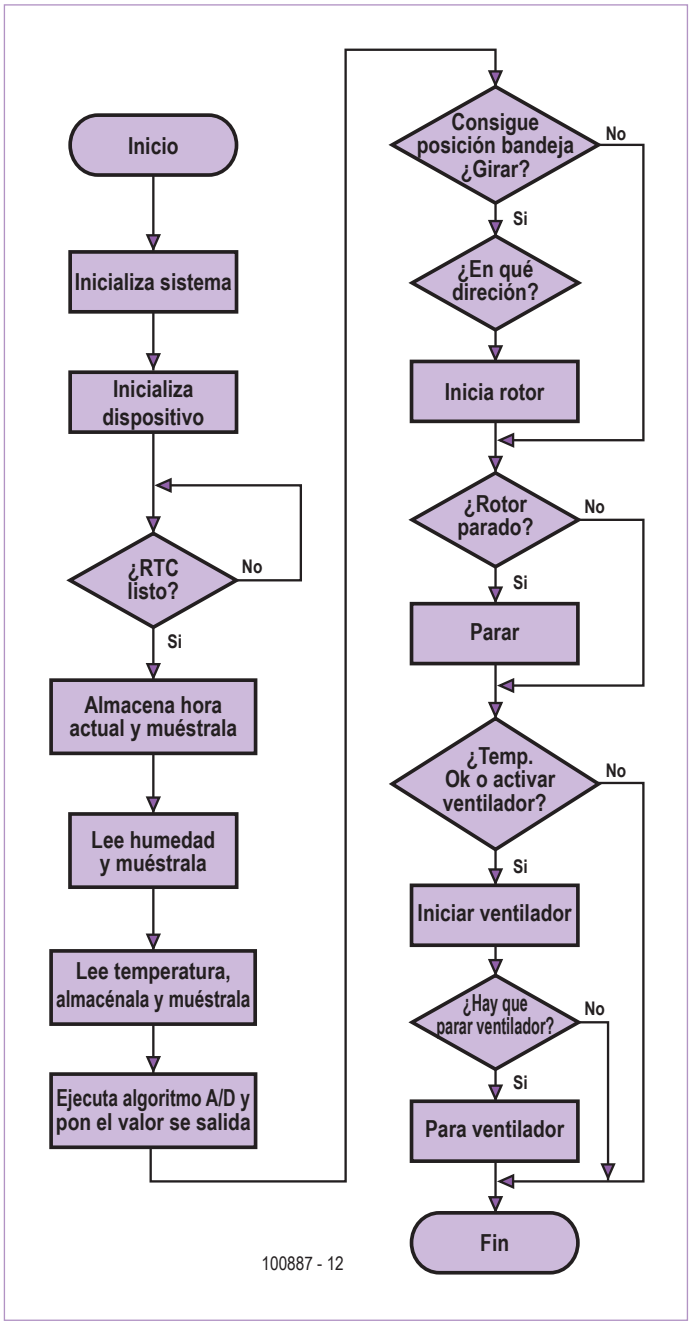


Figura 3. Diagrama de flujo del programa escrito para el microcontrolador LM3S1138 “Stellaris” de TI (Luminary Micro), que gobierna el funcionamiento del sistema de incubación de huevos.

El software

En la **Figura 3** se muestra un diagrama de flujo del programa de control. Todos los aspectos de la construcción mecánica, la electrónica y las limitaciones de mundo real de la incubación de huevos han sido debidamente tenidos en cuenta, en un pequeño trozo de código, para un controlador ARM razonablemente potente. El archivo con el código fuente para el LM3S1138 lo podemos descargar gratuitamente de [2].

(100887)

Enlaces en Internet

[1] http://www.luminarymicro.com/products/LM3S1138.html (hojas de características del LM3S1138)

[2] www.elektor.es/100887 (esquemas eléctricos del circuito y programas del proyecto)

Faro LED para bicicleta

Alta tecnología en caminos campestres

Thomas Finke (Alemania)

Acorde con las normas de tráfico, en el mercado podemos encontrar luces para bicicleta con bombillas halógenas de 2,4 W como máximo. En la oscuridad y alejados de la civilización uno en realidad no puede ver nada de nada con esto. El faro aquí presentado ciertamente no está permitido llevarlo por la calle, pero incorpora cuatro LEDs de alta potencia con unos 600 lumen y puede regularse por medio de un regulador con tan sólo pulsar un botón. La alimentación se lleva a cabo mediante una batería de cuatro células Li-Ion integradas en el propio faro.

La capacidad de la batería de 2,2 Ah por célula basta para iluminar a tope de potencia en campo abierto durante aproximadamente tres horas y en el modo regulado para muchas más. El microcontrolador del circuito se encarga de gestionar la carga, con lo que basta con conectar un simple adaptador de 5 V, o uno específico para el coche que suministre 5 V, para realizar la recarga.

Aclaración

En todo vehículo los faros han de estar regulados de modo que no deslumbren al que viene de frente. En los faros convencionales la óptica está diseñada de modo que el cono de luz está limitado por la parte superior. Una óptica de este tipo para una lámpara de LEDs casera para caminos rurales y de campo no sólo resulta difícil de realizar, sino que además compromete la buena iluminación del suelo. Una vez que las condiciones del terreno lo permitan, con sólo pulsar un botón podremos reducir la potencia del faro. Este "leve destello" minimiza el posible brillo aparte de ahorrar mucha energía, permitiendo un modo mixto y pudiendo realizarse recorridos nocturnos más largos.
Si quiere circular legalmente en una calle normal, según las normas de tráfico deberá utilizar otra iluminación que esté permitida. El faro de LEDs puede utilizarse mientras nos encontremos fuera de zonas de tráfico urbanas.

Funcionamiento

El único elemento de control de la lámpara es un botón. Si lo pulsamos brevemente se enciende, al pulsarlo otra vez más cambiamos entre "luz media" o "luz a tope". Si lo mantenemos pulsado se apaga. Mediante éste botón también se puede regular la luminosidad de la luz intermedia en cuatro pasos. Encendemos el faro, pero esta vez manteniéndolo pulsado prolongadamente, y posteriormente seleccionamos pulsando brevemente el nivel de intensidad deseada. Tras unos instantes el faro volverá a su funcionamiento normal.
La electrónica del faro informa mediante un LED integrado en el botón del estado de carga de la batería, parpadeando periódicamente dependiendo de la tensión de ésta, entre una y cuatro veces (cuatro veces significa que la batería está llena). Si está por debajo de cierto umbral, el faro pasa automáticamente al nivel de intensidad más bajo para prolongar el tiempo de funcionamiento.

Circuito

Los LEDs blancos tienen una tensión en activo de unos 3,5 V, mientras que las baterías de Li-Ion suministran una tensión de entre 2,7 y 4,2 V. Para operar con estos LEDs la tensión de la batería ha de reducirse o elevarse de acuerdo con su estado de carga. Para esto haría falta un converti-

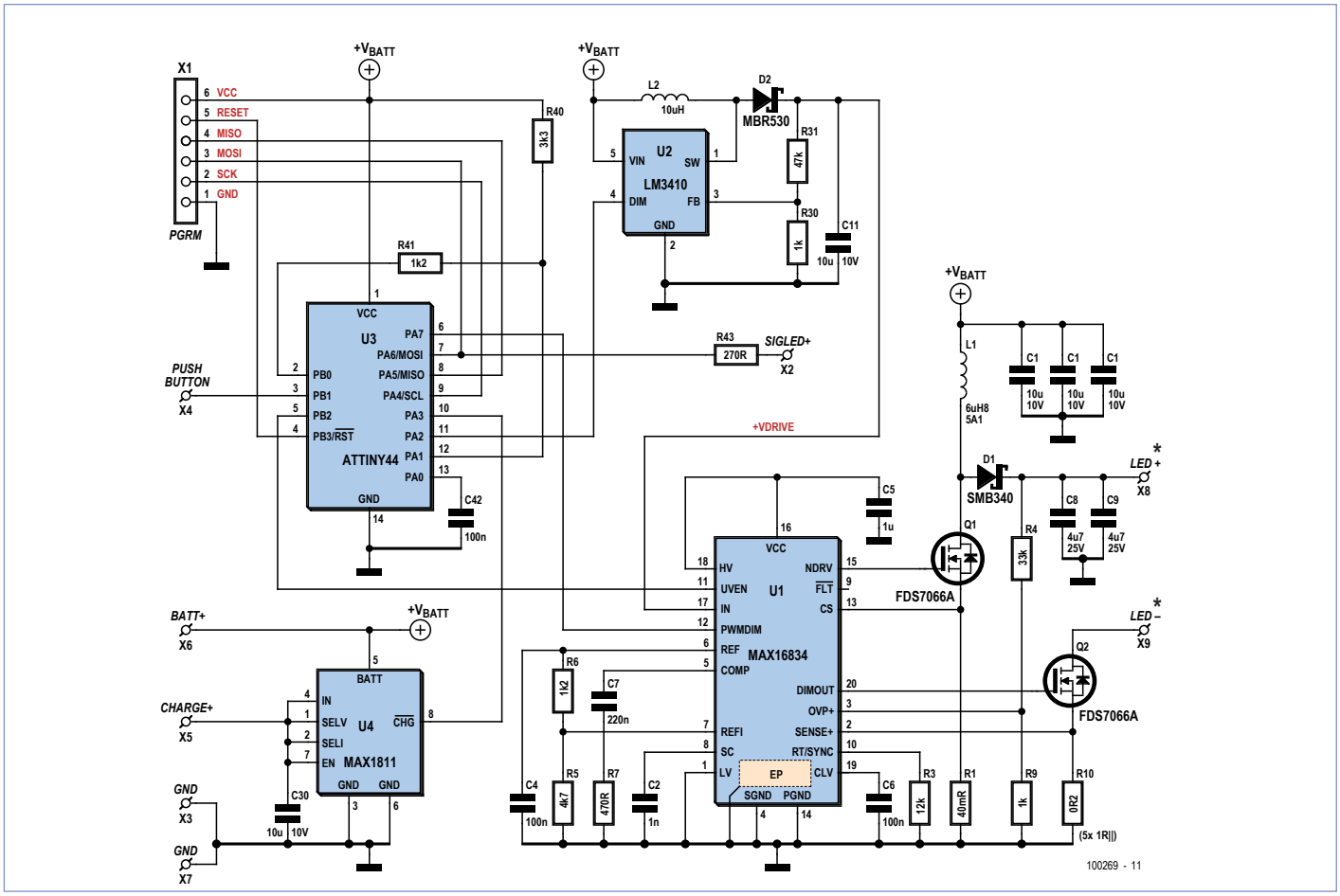


Figura 1. El circuito consta de un microcontrolador, dos convertidores y un regulador de carga para la batería.

dor de corriente continua cuya tensión de salida pudiera ser mayor o menor que la de entrada, como ocurre con los llamados convertidores SEPIC (Single Ended Primary Inductance Converter). El inconveniente en esta clase de convertidor es, aparte de los componentes requeridos, su rendimiento relativamente bajo. Por ello, para este proyecto nos hemos servido de un módulo a modo de tarjeta redonda con cuatro LEDs conectados en serie (CREE XR-E), el cual ya dispone de la óptica necesaria. Posteriormente, para el funcionamiento de los LEDs puede utilizarse un convertidor elevador (Step-up o Boost Converter), que genera la tensión requerida de unos 14 V.

En el circuito (**figura 1**) se ha utilizado un MAX16834 (U1). La conexión del integrado se corresponde esencialmente con el ejemplo de aplicación que puede verse en la hoja de datos, adaptando únicamente los valores de los componentes según nuestro propósito. Hemos de estar atentos a que la bobina L1, con una corriente en los LEDs de 1 A y tensión de batería mínima, debe poder soportar hasta 5,6 A. R10 determina dicha corriente de LED. En la tarjeta, R10 consta de una a cinco resistencias de 1 Ω conectadas en paralelo. Esta corriente depende del número de resistencias de 1 Ω que conectemos. Si montamos cinco, obtenemos un valor para R10 de 0,2 Ω. La corriente de LED puede ser de hasta 1 A. En su prototipo el autor sólo ha utilizado tres resistencias de 1 Ω, limitando la corriente de LED a 600 mA. La luminosidad es más que suficiente y el calor disipado por los LEDs se mantiene en un rango aceptable.

La señal PWM del pin PWMDIM puede utilizarse para regular los LEDs. En este caso se utiliza para reducir la potencia consumida por el faro.

En el pin 11 (UVEN) normalmente se conecta un divisor de tensión que desactiva el componente en caso de que la tensión esté por debajo de un valor aceptable y activa el modo de ahorro energético. En el circuito del faro LED se encarga de esto el microcontrolador Atmel (U3) mediante un pin de puerto (PB2).

Para su funcionamiento y conmutación segura, los MOSFETs, U1 necesita una tensión de funcionamiento de al menos 5 V, para lo cual la tensión de batería no basta. Por ello se utiliza un convertidor adicional basado en un LM3410 (U2), que suministra una tensión estabilizada (+VDRIVE). Realmente, el LM3410 es un regulador de corriente para LEDs de hasta 500 mA. En un principio el autor consideró incluso diseñar el circuito completo del faro con varios de estos componentes. Sólo ha sobrevivido uno, pero dota al circuito de un excelente convertidor elevador minimalista con tensión de salida constante.

El microcontrolador ATtiny44 de Atmel (U3) se encarga de controlar el resto de componentes del circuito. Genera la señal PWM necesaria para regular los LEDs de potencia y pone a U1 y U2 cuando sea necesario en modo de standby. El conversor analógico-digital integrado supervisa la tensión de la batería mediante el divisor de tensión formado por R40 y R41. Para que este divisor de tensión no descargue la batería innecesariamente, R41 no está conectado directamente a masa, sino a PB0. Este pin del puerto puede fijarse

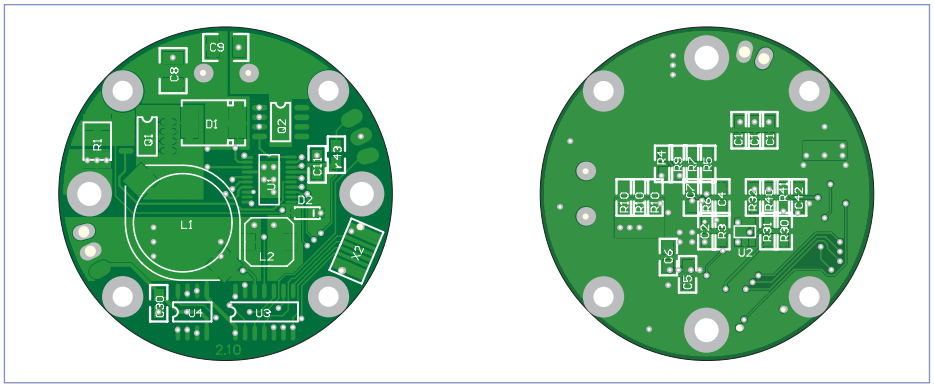


Figura 2. Para su montaje se ha diseñado una tarjeta con forma circular.

Figura 3. Tarjeta redonda montada con SMDs.

Figura 4. Vistazo al interior del faro.

como alta impedancia, interrumpiendo así el flujo de corriente por el divisor de tensión.
En realidad el faro nunca está completamente apagado. En lugar de eso, el micro Atmel, el convertidor auxiliar y el driver LED se ponen en modo standby. En conjunto consumen unos 100 µA, lo cual podemos asumir teniendo en cuenta que la capacidad de la batería es de 8,8 Ah. Para despertarlos, el botón dispara en PB1 una interrupción en el controlador.

Alimentación

De la carga de la batería se encarga el MAX1811 (U4). Este componente de Maxim se diseñó originalmente para cargar baterías de Li-Ion a través del puerto USB. Este integrado suministra únicamente una corriente de carga de unos 500 mA, una carga completa podría durar hasta unas 20 horas. Pero en cualquier caso, sería un problema alcanzar corrientes de carga más altas con los conectores utilizados, aparte de tener que supervisar la temperatura de la batería por motivos de seguridad. Ya que el faro sólo se va a utilizar durante unas pocas horas al día y por ello es casi seguro que sólo se descargue parcialmente, a la hora de su funcionamiento práctico esto no supone limitación alguna.
La batería en sí consta de cuatro células circulares conectadas en paralelo, procedentes de un pack de un ordenador portátil estropeado. Si se hubieran conectado las células en serie ciertamente nos podríamos haber ahorrado el convertidor auxiliar, pero también hubiese aumentado la complejidad de la carga y hubiésemos tenido que balancear las células.
Con la corriente de LEDs limitada a 600 mA del prototipo basta una carga para disponer de un funcionamiento continuo durante 3,5 horas a máxima intensidad.

Montaje

Para la construcción del circuito se ha diseñado una tarjeta redonda de doble cara (**figura 2**). La tarjeta montada puede verse en la **figura 3**.
En los últimos años, el rendimiento de los LEDs no ha parado de mejorar. No obstante, durante su funcionamiento todavía se irradian grandes cantidades de calor. El módulo LED (4 x CREE XR-E sobre una tarjeta circular con 34 mm de diámetro) está montado sobre una lámina de aluminio de 5 mm de grosor, que hace buen contacto con el resto del encapsulado, de modo que conduce bien el calor al entorno. La parte posterior de la lámina hace al mismo tiempo de disipador para los MOSFETs del circuito regulador.
La propia carcasa está hecha de forma circular, con un trozo de tubo de 50mm de diámetro taladrado y anodizado, lo cual sorprendentemente no ha dado ningún problema. En la web de este artículo [1] puede encontrar dos bocetos relativos al montaje y construcción de la carcasa. En la **figura 4** podemos ver el montaje del faro: frente de la carcasa con la lente, tarjeta de LEDs, lámina de 5 mm de grosor como disipador, tarjeta con electrónica, separador, lámina de plástico y células de la batería.
El encapsulado ha de ser totalmente estanco al agua. Para ello, la lente se fija con silicona en el frente de la carcasa, y tanto por delante como por detrás utilizamos anillos en O como sellado en el tubo. El botón es del tipo resistente al agua con un LED integrado y parte posterior completamente sellada con pegamento. El conector del cargador es del tipo SMB, por motivos de seguridad. Dicho conector coaxial en miniatura está chapado en oro, resultando así inmune a la corrosión. Éste también se fija a la carcasa con pegamento para que sea resistente al agua.
Gracias a estas medidas de montaje, la lámpara es capaz de funcionar diariamente durante casi dos años, resultando finalmente imprescindible en los caminos más difíciles.

(100269)

[1] Página del proyecto de lector con las descargas disponibles: www.elektor.es/100269

Desarrollar para Android

sobre PC, BeagleBoard, teléfono o tarjeta

Clemens Valens (Elektor Francia)

A finales de 2008 apareció el primer teléfono móvil que ejecutaba el sistema operativo Android. Ahora el número de teléfonos móviles Android activados cada día está estimado en unos 350.000... ¿Cuál es la razón de este éxito tan enorme? ¿Google? ¿Porque es de «código abierto»? ¿Porque es bueno? Poco importa la razón, lo que aquí cuenta es que ¡nosotros también podemos «hacer» Android y estar en el *top* de las tendencias!

¿Qué es Android?

A finales de 2003 se fundó la sociedad Android Inc., en Palo Alto, California, en los Estados Unidos, cuyo objetivo era el de desarrollar programas para teléfonos móviles. Apenas dos años más tarde, la empresa era comprada por Google. Cuatro años después del nacimiento de Android Inc, ya estamos a finales de 2007, nació la *Open Handset Alliance*, de la que Google forma parte. Esta alianza se puso como objetivo el desarrollo de los estándares abiertos para dispositivos móviles. Su primer producto es Android, una plataforma para dispositivos móviles basada en el núcleo (*kernel*) 2.6 Linux. Un año más tarde era comercializado el primer teléfono móvil que ejecutaba ya este nuevo sistema operativo, el HTC Dream.

Android no es Linux

Incluso si en su origen Android está basado en Linux, hoy se ha convertido en un sistema operativo (SO) de pleno derecho. Los códigos fuentes de Android siempre son abiertos, pero ya no forman parte del *código base* de Linux. Google modificó ciertos aspectos de Linux de modo que Android no ofrece un sistema de ventanas "X Windows" y todas las bibliotecas estándar de GNU ya no están presentes. Así pues, la exportación de aplicaciones Linux existentes sobre Android es bastante difícil. Además, Google ha añadido sus propias funciones, particularmente las referentes a la seguridad de los dispositivos móviles.

Otra divergencia de Linux es la licencia de Android. Linux es distribuido bajo la licencia GPL de GNU, mientras que Android es distribuido bajo la licencia Apache de la ASF (*Apache Software Foundation*). Esta licencia, contrariamente a la licencia GPL, autoriza la distribución de programas propietarios basados en programas libres (distribuidos bajo la misma licencia, por supuesto) sin divulgar los códigos fuentes.

Android tampoco es Java

Las aplicaciones (las *apps)* para Android están escritas en Java, pero no son ejecutadas como aplicaciones Java, ya que Android no integra ni una máquina virtual Java ni las bibliotecas Java: así pues, Android no sabe ejecutar programas Java. Las aplicaciones Android, que utilizan solamente la sintaxis del lenguaje Java, son ejecutadas por una máquina virtual *Dalvik*, un tipo diferente de Java. ¿Nos seguís?

Por otra parte, es totalmente posible desarrollar bibliotecas para Android utilizando otros lenguajes de programación, como C o C ++. Este tipo de bibliotecas, al igual que sucede con las DLLs en Windows, pueden ser importadas y utilizadas por Dalvik.

Hacer Android

Cada uno puede desarrollar sus aplicaciones para Android. Todas las herramientas para hacerlo son gratuitas, basta con descargarlas, ni siquiera es necesario tener acceso a una plataforma Android. En efecto, en el sitio para desarrolladores de Android [1], encontramos todo lo que necesitamos, incluido emuladores de material Android. El kit de desarrollo software, SDK *(Software Development Kit)*, está disponible para Windows, Linux y Mac OS X.

Incluso si Android solo utiliza la sintaxis de Java, su SDK utiliza Java a fondo. Por lo tanto, hay que instalar también el kit de desarrollo Java (JDK) y, por supuesto, disponer de una máquina virtual de Java (JRE). El entorno de desarrollo integrado (EDI) recomendado es Eclipse, aumentado con el "injerto" ADT (*Android Development Toolkit*).

La instalación del SDK (con JDK, Eclipse, ADT, etc.) está muy bien explicada en su página de Internet, por lo que no voy repetirla aquí. La única cosa que hay que saber antes de lanzarse es que la instalación, o más bien la descarga de todos los

componentes, puede llevarnos fácilmente unas horas, según la velocidad de nuestra conexión Internet. En cualquier caso, estamos hablando de varios GBytes.
Una vez que tenemos todo instalado, se aconseja probar el tutorial HelloAndroid. Aquí también nos encontramos con todo bastante bien explicado salvo, tal vez, la elección de la versión de Android. En efecto, hay varias versiones de Android, con nombres que evocan postres: versión 1.5 *Cupcake* (un tipo de pequeño bizcocho), 1.6 *Donut*, 2.0/2.1 *Eclair*, 2.2 *Froyo* (de una marca de yogur helado), 2.3 *Gingerbread* (pan de especia) y 3.0 *Honeycomb* (panales de miel). La versión de Android determina el circuito (o teléfono) que podrá ejecutar nuestras *app*. La versión 3 está prevista para las "Tablets PC" (o "tableta"), la versión 2.2 (Froyo) es actualmente la más habitual (al parecer, también es posible instalarla sobre un iPhone o un iPod táctil), seguida por Eclair (2.1). Así pues, elegiremos la versión que corresponde a nuestro material.
Seguidamente, hay que crear y arrancar un periférico virtual Android (*Android Virtual Device* o AVD) compatible con la versión Android elegida. Es sobre este periférico en el que podemos probar nuestra aplicación antes de instalarla sobre un verdadero dispositivo Android. El arranque del AVD es bastante lento (según nuestra máquina de desarrollo) y puede mostrarse "goloso" en consumo de memoria. Cuanto más alta es la versión de Android, más lento es su AVD. Una "tableta" virtual (es decir, un Android versión 3) tardó varios minutos antes de estar operativo sobre mi máquina de prueba (Windows XP SP3, 2 GB de RAM, Pentium T4200 @ 2 GHz). ¡Paciencia pues!

La programación para Android: ¿un juego de niños?

¿No sabemos programar pero, a pesar de todo, queremos realizar una aplicación para nuestro teléfono o tableta Android? Esto es posible gracias a *App Inventor for Android* (¡Inventor de aplicaciones para Android!) [2]
Necesitamos un ordenador con un sistema operativo reciente (Windows, Linux, Macintosh), un navegador (con conexión) de Internet y Java. Antes de ponernos nerviosos inútilmente, verificaremos en línea, sobre el sitio de *App Inventor*, la compatibilidad de nuestra máquina de desarrollo. Si todo funciona como es debido, descargaremos *App Inventor Setup* (casi 100 MB) y lo instalaremos sin modificar las opciones propuestas.
La etapa siguiente es la conexión del dispositivo Android. Es aquí donde esto se complica un poco, ya que necesitamos un controlador para el dispositivo y vamos a tener que buscarlo en Internet, excepto si poseemos uno de los teléfonos reconocidos (por ejemplo, Nexus One).
Sí, como yo, nuestros lectores no disponen (todavía) de un teléfono Android, podemos, a pesar de ello, comenzar a desarrollar, ya que *App Inventor* contiene un emulador. Lanzar el emulador puede llevarnos algo de tiempo, hasta unos minutos, según Google, pero sobre la máquina de prueba, el emulador estuvo operativo en unos 30 s.
El procedimiento para crear una aplicación es idéntico para los dispositivos Android reales y para los virtuales. Primero, abrimos *App Inventor* haciendo clic sobre *My Projects* desde la página inicial (suponiendo que estamos conectados con nuestra cuenta Google). La ventana que se abre se llama *Designer (Diiseñador)* (ver **Figura 1**). Es aquí donde se diseña la parte visible de la aplicación, colocando objetos como botones, imágenes, sonidos, etc. sobre una superficie que corresponde a la pantalla del dispositivo Android. Modificando ciertas propiedades de los objetos, podemos personalizar la apariencia de la aplicación.

Figura 1. Android App Inventor: *Diseñador*.

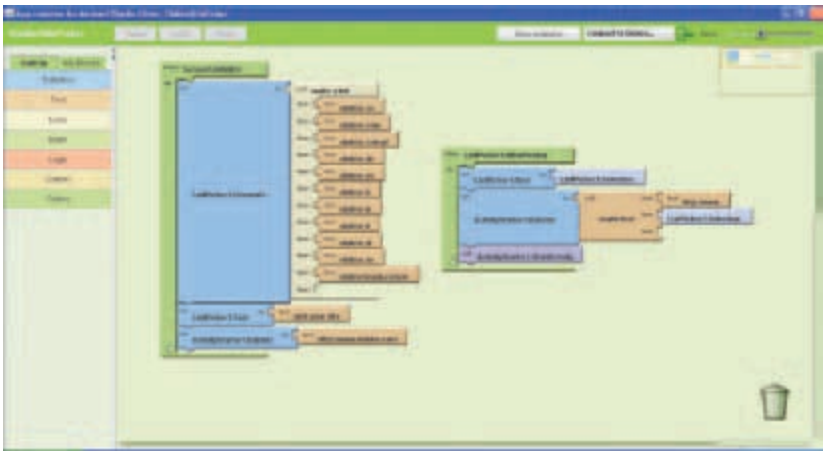


Figura 2. La aplicación de prueba en *Blocks Editor*.

Atención: ¡trampas!

Me encontré con diferentes problemas durante la instalación de Android sobre la BeagleBoard. El primero se manifestó rápidamente, cuando quise lanzar el "script" de instalación **mkmmc-android.sh**. Se produjo un error y sólo se creó la primera partición en mi tarjeta SD. En un momento de lucidez, me di cuenta de que el "script" efectuaba una búsqueda con *grep* de la palabra "Disk" y, como mi Ubuntu estaba en francés, ¡había, pues, que buscar la palabra "Disque"! En lugar de modificar el "script", cambié el idioma de Ubuntu para evitar dificultades similares con otros "scripts". Si modificamos el "script" y si lo guardamos después con otro nombre, debemos recordar el hacerlo ejecutable (clic derecho-> *properties*-> *permissions*).
El segundo problema tenía que ver con el vídeo. La tarjeta SD proporcionada con la BeagleBoard contiene la distribución Ångström, una versión de Linux adaptada a los sistemas embebidos. Cuando arranqué la BeagleBoard con este SO, todo iba bien y obtenía, efectivamente, una imagen sobre mi pantalla TFT. En cambio, al tratar de arrancar Android o Ubuntu acababa en una pantalla negra, incluso si la pantalla funcionaba perfectamente con mi ordenador. La causa del problema residía en la resolución de la señal vídeo, incompatible con mi pantalla. En efecto, las distribuciones de Android y de Ubuntu para la BeagleBoard parece que utilizan, por defecto, una resolución algo más extraña de 1280 x 720 pixeles, un modo que mi pantalla no tolera.
Para corregir el problema, hay que modificar el fichero binario **boot.scr**, algo que puede hacerse con ayuda de la herramienta **mkbootscr** proporcionada en el SDK de TI, en el directorio *Tools*. El inconveniente está en que este "script" sólo produce un fichero **boot.scr** para otro tipo de placa. Afortunadamente, el "script" crea también un fichero **boot.cmd** que se puede modificar con un editor de texto antes de transformarlo en boot.scr.
En el fichero **boot.cmd**, reemplazamos los **bootargs** por los descritos en el guía de usuario del SDK de TI, después, añadimos al final, pero antes del carácter «□» **omapfb.mode=dvi:1024x768MR-16@60**, sustituyendo la resolución 1024x768 por otra resolución que sea reconocida por nuestra pantalla. Ahora creamos el fichero **boot.scr** con el comando **mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n 'Execute uImage.bin' -d boot.cmd boot.scr** y lo copiamos en la tarjeta SD, en la raíz de la partición *boot*.
Para el fichero **boot.scr**, la nota de TI especifica una frecuencia de reloj de 1 GHz (**mpurate=1000**), pero buscando un poco en internet, encontramos que la BeagleBoard no trabaja bien a frecuencias demasiado elevadas. No lo intenté con esta frecuencia y preferí utilizar **mpurate=800**.
Mi fichero **boot.scr** que funciona está disponible en [4].
La última trampa que hay que evitar es la de los derechos de usuario cuando se crea la tarjeta SD manualmente (algo que se puede hacer perfectamente). Este proceso hay que hacerlo obligatoriamente siendo usuario **root**, si no, Android no consigue acabar su trabajo y un terminal conectado al puerto serie de la BeagleBoard nos mostrará mensajes que contendrán las palabras *Permission denied* (*Permiso denegado)*. Para hacernos usuario root en Ubuntu abriremos una ventana de comandos (terminal) y escribimos **sudo-s**. Ahora, podemos preparar la tarjeta SD manualmente a partir de este terminal. No debemos olvidar que la partición *boot* de la tarjeta SD debe ser una partición de arranque. Tampoco debemos olvidar ejecutar el comando **/media/rootfs# chmod-R 755 / mnt** para autorizar el acceso al sistema de ficheros *rootfs*.

El comportamiento de la aplicación se define en el *Blocks Editor* (*Editor de Bloques*). La programación es presentada como una especie de juego de construcción, donde las propiedades de los objetos y las acciones toman formas que hacen pensar en piezas de un rompecabezas. Ensamblando las formas creamos un programa (ver **figura 2**). Este modo de proceder se parece mucho al Scratch [3], una técnica de programación concebida, cito, *«para iniciar a los niños, a partir de 8 años, con conceptos importantes en matemáticas e informática, aprendiendo a desarrollar un pensamiento creativo, un razonamiento sistemático y a trabajar en equipo.»* Así, programar para Android de esta forma sería un juego de niños y debería, pues, estar al alcance de un lector de Elektor no demasiado entumecido.
Programar así permite, en efecto, construir rápidamente una pequeña aplicación, pero no podemos hacer todo, y todo no es necesariamente fácil. Además, *App Inventor* está aún en su fase "beta" y es posible encontrar *bug*. (errores de la aplicación aún no corregidos). Durante mis pruebas, por ejemplo, tuve grandes problemas con el sintetizador de voz cuando la aplicación contenía también un objeto *ActivityStarter*.
Una vez finalizada la aplicación, podemos "embalarla" *(Package for Phone)* y descargarla, sea sobre el ordenador, o bien directamente sobre el dispositivo Android. Si el teléfono no está conectado al ordenador, también podemos obtener un «*flash code*» (código QR, ver **Figura 3**), que contiene el enlace a la aplicación y, así, descargarlo sobre el teléfono (a condición de que este

¡Esto funciona! Aquí tenemos la aplicación de prueba sobre un verdadero teléfono Android.

último sepa leer los códigos flash, por supuesto, y que esté conectado a Internet). También es posible recuperar los códigos fuente de la página de gestión de los proyectos. Podéis descargar mis experiencias en [4] e importarlas en vuestro proyecto.

La parte material

En cuanto al especialista en electrónica, ¿tal vez os gustaría realizar, por vosotros mismos, el circuito capaz de ejecutar Android? y, si es así, ¿cuáles son las especificaciones? Pues bien, no está muy claro, pero creo que lo mínimo indispensable sería un procesador ARM trabajando a 200 MHz, con 32 MB de memoria RAM y 32 MB de memoria Flash. No obstante, se recomienda un mínimo de 128 MB de RAM y 256 MB de Flash. Android 3 necesita un procesador que trabaje a una frecuencia de 1 GHz, más 512 MB de RAM. Como periféricos, necesitaremos un puerto USB, una pantalla TFT QVGA de 65.536 colores o mejor, así como diez teclas. Elementos útiles, aunque no obligatorios, son un lector de tarjetas (micro) SD, una cámara de fotos de 2 megapíxeles y Bluetooth.
Está claro que podemos realizar un sistema así por nosotros mismos, pero es más simple y, probablemente menos caro, comprar una placa totalmente preparada. Buscad un poco en Internet y seguro que encontraréis algo no muy caro, por ejemplo la...

BeagleBoard

Hace unos años, Texas Instrumentos (TI) desarrolló una placa denominada BeagleBoard [5] para dar valor a sus procesadores multimedia OMAP. La placa es un "hardware abierto" y mantenida por la comunidad de "código abierto". Incluso si los esquemas eléctricos y los planos de montaje de la placa están disponibles gratuitamente, tendremos problemas para construir por nosotros mismos la BeagleBoard, ya que la memoria RAM (¡512 MB!) está montada *sobre* el procesador (PoP, *package on package*, encapsulado sobre encapsulado). Afortunadamente, la placa no es muy cara (la versión xM cuesta sólo 110 €) y es fácil de obtener.
La BeagleBoard ofrece muchas cosas, sobre la xM, la última versión (ver **figura 4**), encontramos (entre otros) cuatro puertos USB (huésped y periférico), un puerto Ethernet, un puerto RS-232, un puerto USB OTG, un conector micro SD, una entrada/salida estéreo de audio, una salida S-Vídeo, un puerto HDMI y conectores de extensión para una pantalla LCD, cámara y montajes personales. El procesador o, para ser más exacto, el sistema de circuito integrado único (SoC) es un DaVinci DM3730 *Digital Media Processor*, que integra, entre otros, un núcleo ARM Cortex-A8, un DSP TMS320C64 y un acelerador gráfico SGX. El procesador es compatible con numerosos SO como Windows CE, Linux, QNX y, también, Android.
Contrariamente a la primera versión de la BeagleBoard, la versión xM no posee memoria flash. La memoria RAM está complementada aquí por una tarjeta micro SD, sobre la que se han instalado el SO y las aplicaciones. La ventaja de esta configuración: es muy fácil cambiar el SO, ya que basta con cambiar la tarjeta SD. El inconveniente es una pérdida de prestaciones, ya que las tarjetas SD no son famosas por su rapidez. La utilización de las memorias USB para ciertas cosas, permitiría mejorar un poco dichas prestaciones.

Figura 3. El código flash para acceder a la aplicación desde un teléfono móvil.

La BeagleBoard es, pues, un pequeño ordenador polivalente y potente, utilizable para toda clase de aplicaciones. Incluso si, claramente, no es el mejor "alojamiento" para Android, lo escogí (a pesar de todo) como plataforma para mis experimentos ya que yo no disponía ni de teléfono ni de tableta Android. Hay que completar la plataforma de experimentación con:

- una alimentación externa de 5 V, capaz de proporcionar 1 A sin bajar la tensión (observé un consumo de casi 3,5 W, es decir, 700 mA con 5 V, cuando la placa ejecutaba Android).

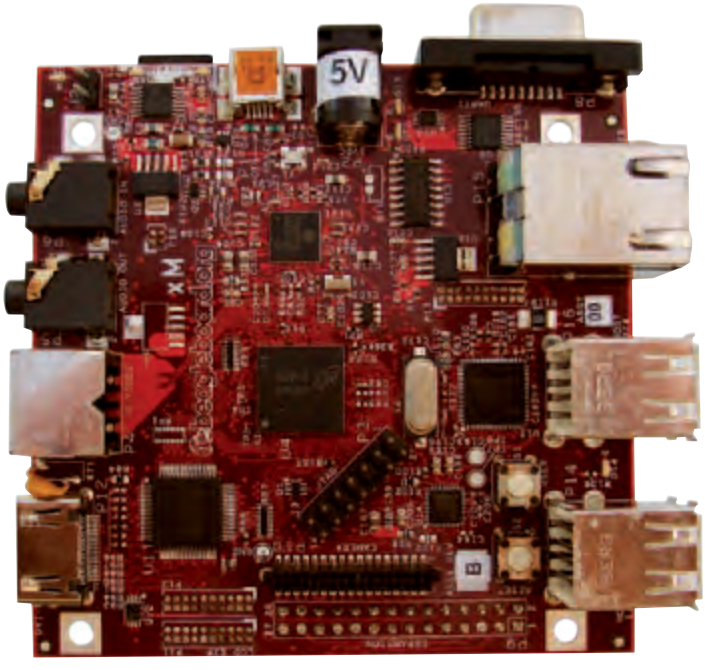


Figura 4. La BeagleBoard-xM está alimentada con 5 V. Consume casi 700 mA cuando ejecuta Android.

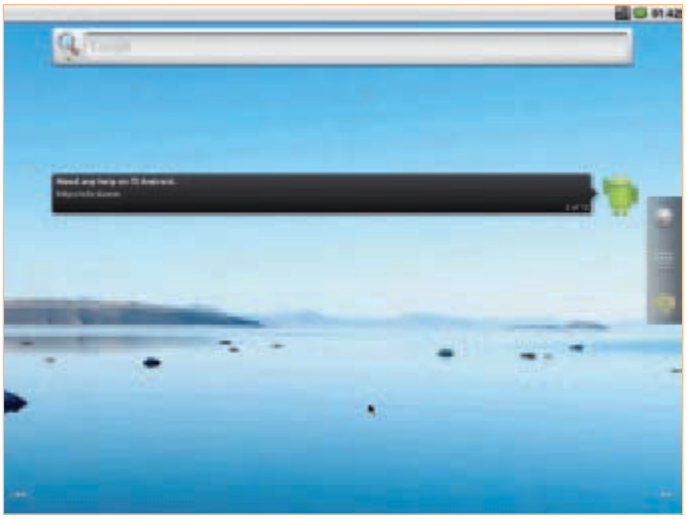


Figura 5. La pantalla de bienvenida de Android sobre BeagleBoard.

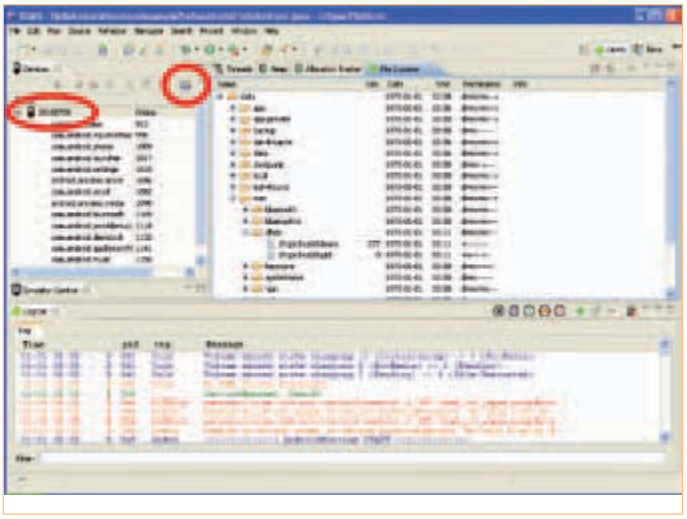


Figura 6. Vista de DDMS en Eclipse. Nuestro periférico Android se llama «20100720». La pequeña cámara de fotos permite tomar fotos desde nuestra pantalla.

- una pantalla con entrada DVI-D (incluso si la BeagleBoard ya dispone de un conector HDMI, éste ofrece sólo las señales DVI-D).
- un concentrador (*hub*) USB con alimentación externa (ya que los puertos USB de la placa no pueden proporcionar corriente suficiente)...
- sobre los que conectar un teclado y un ratón USB.

Para instalar Android (o Ubuntu), recomendamos una tarjeta micro SD de, por lo menos, 4 GB. La BeagleBoard viene con una tarjeta de este tipo pero, para no destruir la distribución Ångström (una distribución Linux para sistemas embebidos) preinstalada sobre esta tarjeta, me hice con una tarjeta SD suplementaria de 8 GB.

TI pone a disposición del público todo lo que se necesita para arrancar con Android pero hace falta, obligatoriamente, un ordenador (virtual) bajo Linux, con lector de tarjeta SD para preparar la tarjeta SD. TI recomienda Ubuntu 8.04 (o una versión más reciente), y es lo que yo he hecho, usar un Ubuntu 10.04. Después de la descarga del **TI_Android_FroYo_DevKit-V2.2** (1,1 Go) [6], podemos comenzar. El procedimiento es muy simple, en principio, pero podemos encontrar algunos problemas. Recomendamos leer el apartado «**Atención: trampas**» si el proceso no se desarrolla como es debido.

Sobre la máquina Ubuntu, abrimos una ventana de comandos (con alt-F2, por ejemplo, elegimos la opción Run in *terminal*) y tecleamos **sudo-s** para poder ser usuario **root** con todos los derechos posibles. Esto que viene ahora no es totalmente necesario, ya que podemos preparar la placa con la ayuda de **sudo**, pero es mucho más práctico en caso de problemas. Descomprimimos el archivo **tar-xzvf TI_Android_FroYo_DevKit-V2.2.tar.gz** y vamos, con la ayuda del comando **cd**, al directorio **TI_ANDROID_FROYO_DEVKIT-V2.2\PREBUILT_IMAGES\BEAGLEBOARD-XM**, que se encuentra, a partir de ahora, en alguna parte de nuestro ordenador. Desde este directorio ejecutamos el "script" **mkmmc-android** con el comando (precedido por **sudo**, si no estamos en root) **./mkmmc-android.sh / dev / sd <Device>**. Aquí, debemos reemplazar <device> por el "nombre" del lector de tarjetas. En mi caso era «c», con lo que queda **/dev/sdc.** Hay varias técnicas para descubrir el nombre del lector, yo lo localicé con la ayuda de la herramienta de disco de Ubuntu (System-> Administration). Mejor no equivocarse, ya que el "script" de instalación de Android formateará el disco que especifiquemos.

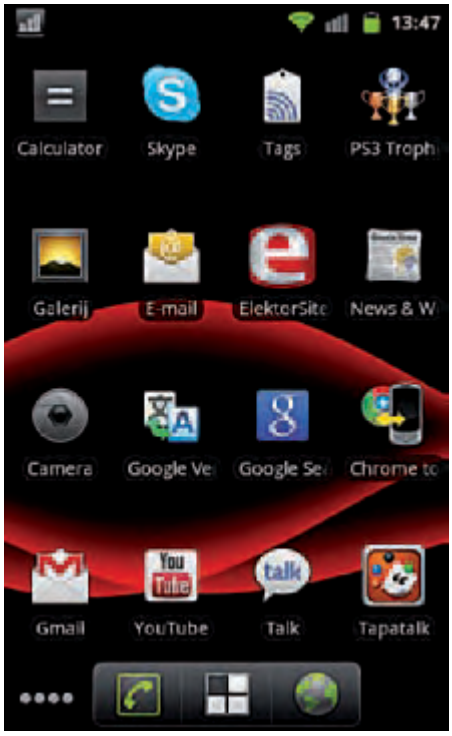
La ejecución del "script" lleva un poco de tiempo pero, si no encuentra problemas, ya habremos acabado con Ubuntu. A partir de ahora, sobre la tarjeta SD se encuentran tres particiones: *boot*, *rootfs* y *data* (con un tamaño de, respectivamente, 74 MB, 4 GB y 3,9 GB, sobre una tarjeta SD de 8 GB). Colocaremos la tarjeta SD en el lector de la BeagleBoard y arrancaremos el sistema. La primera vez tarda bastante (más de un minuto en mi caso), pero qué felicidad el ver finalmente aparecer sobre la pantalla la palabra Android, seguida por la pantalla de bienvenida de Android (ver **Figura 5**). Es instructivo conectar un terminal sobre el puerto RS-232 (parámetros del puerto: **115200n81**) de la BeagleBoard para controlar los mensajes que desfilan durante el arranque.

No debemos olvidar nunca que Android está pensado para sistemas con pantalla táctil y que todo IHM (Interfaz Hombre-Máquina) está basado en gestos (golpes y deslizamientos). Si vuestra pantalla, como la mía, no es táctil, utilizad el ratón como dedo (golpee con un clic del botón izquierdo, deslícese manteniendo pulsado el botón izquierdo, clic en el derecho significa *volver*). Podemos hacer casi todo con un ratón excepto abrir el menú (en realidad, no he encontrado cómo hacerlo). No es muy molesto en sí mismo, excepto si, como yo, deseamos cambiar el fondo de pantalla (no encontré cómo hacerlo de otra forma). Para acceder al menú, pulsamos, pues, sobre la tecla **F1** del teclado. Por otro lado, es la única tecla que utilicé durante todos mis experimentos con Android.

Una vez que mi BeagleBoard ya era capaz de ejecutar correctamente Android, abandoné Ubuntu para continuar sobre Windows XP, aquí me siento más a gusto. Otra razón para esta elección es la falta de pantallas, ya que me vi obligado a compartir mi única pantalla TFT entre Ubuntu y Android, algo que terminó haciéndose un poco penoso. Windows XP se ejecuta sobre mi ordenador portátil que tiene su propia pantalla.

La siguiente etapa es la de integrar la BeagleBoard en Android SDK (si aún no lo hemos instalado, debemos ir al apartado «**Hacer Android**», explicado más arriba y hacerlo ahora). Para el cómo se hace realmente, sólo podemos decir que todo depende de nuestro SO donde se instala y su funcionamiento está bien explicado en la nota de TI.

¡Estupendo! Nuestro icono está entre los otros.



Como yo he continuado bajo Windows, he aquí cómo hacerlo para Windows.
Descargaremos y descomprimiremos el fichero **usb_driver_r03-windows.zip** [7]. Abrimos una pantalla de comandos y escribimos (seguido de <Enter>, por supuesto):

```
echo 0x18D1> "%USERPROFILE%\.android\adb_usb.ini"
```

Este comando críptico crea un fichero de texto llamado *adb_usb.ini* que contiene el texto "0x18D1" en el directorio **.android** del directorio donde Windows almacena nuestros parámetros personales. (Teclee **echo %USERPROFILE%** para descubrir dónde exactamente.)
Seguidamente abrimos en el Bloc de Notas el fichero **android_winusb.inf**, contenido en el directorio del controlador USB que acabamos de descomprimir y, bajo el encabezamiento **[Google. NTx86]**, añadimos las líneas siguientes:

```
; Beagle Board
%SingleAdbInterface% = USB_Install, USB\VID_18D1&PID_9018
%CompositeAdbInterface% = USB_Install, USB\VID_18D1&PID_9018&MI_01
```

Ahora, salvamos y cerramos el fichero.
Arrancamos la placa BeagleBoard y esperamos a que la pantalla de bienvenida de Android aparezca. Conectamos entonces un cable USB entre el ordenador y el conectador **mini USB** de la BeagleBoard, el conector situado al lado de la toma de alimentación, y ayudamos a Windows a instalar el controlador. Hay que hacerlo a mano, dando a entender a Windows que el controlador se encuentra en un lugar sólo conocido por nosotros. Guía a Windows hasta el fichero **android_winusb.inf** modificado y deja que Windows acabe su trabajo.
Una vez que Windows reconozca la placa BeagleBoard como un *Android ADB interface* (desconecta y vuelve a conectar si es necesario), podremos hacer una prueba de conexión en una ventana de comandos, escribiendo los comandos en negrita:

```
adb kill-server
adb start-server
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
adb devices
List of devices attached
20100720 device
```

Esto quiere decir que nuestra BeagleBoard ha sido reconocida como periférico ADB bajo el nombre (dulce) de "20100720" y que, en lo sucesivo, podremos utilizarlo como dispositivo Android en el *App Inventor* o en *Eclipse*. Se nos propondrá este nombre o número cuando tratemos de ejecutar la aplicación que estamos desarrollando.
Con la ayuda de esta ventana de comandos, podemos instalar (**adb install mon_appli.apk**) o desinstalar (**adb uninstall mon_appli.apk**) las aplicaciones que hayamos descargado (de Android Market, por ejemplo). La desinstalación puede hacerse también directamente en Android, gracias al gestor de aplicaciones.
Con el comando **adb shell**, obtenemos una ventana de comandos para Android en la cual podemos, por ejemplo, manipular ficheros con la ayuda de los comandos Linux.
En Eclipse, disponemos de una vista *(perspectiva)* **DDMS** (*Dalvik Debug Monitor Server*) para (entre otros) tomar fotos de la pantalla de nuestro periférico Android, pulsando sobre la pequeña cámara de fotos (**Figura 6**). Esta herramienta está también disponible fuera de Eclipse. Se encuentra en el directorio ***tools*** de nuestra instalación del SDK, bajo el nombre (sí, lo adivinasteis), **ddms.bat**.
Para conectar a Internet la BeagleBoard con Android, hay que saber que el puerto Ethernet, al igual que los cuatro puertos USB de la placa está, de hecho, conectado a un "hub" (concentrador) USB integrado en la placa. Por consiguiente, el puerto Ethernet no se llama **eth0**, sino **usb0** y habrá pues que sustituir *eth0* con *usb0* en aquellos comandos que conciernan a la red. Conectamos la placa BeagleBoard a nuestra red y, en el ADB shell, ejecutamos el comando **netcfg usb0 dhcp** para obtener una dirección IP (suponiendo que nuestro servidor DHCP está en línea). Recuperamos la IP de nuestro DNS (**getprop net.usb0.dns1 usb0.dns1**) y le decimos a Android que tendrá que utilizarla si quiere tener acceso a Internet (**setprop net.dns1 dns1 <IP obtenida>**). Y, ¡ya está! ¡Nuestra BeagleBoard está en línea!
Creo que, de aquí en adelante, disponemos de todas las informaciones necesarias para arrancar como desarrolladores de Android. Cuando hayáis acabado vuestra primera aplicación para controlar por Bluetooth un montaje personal, por favor enviádmela.

(110265)

Enlaces en internet

[1] http://developer.android.com/index.html
[2] http://appinventor.googlelabs.com/about/
[3] http://scratch.mit.edu/
[4] www.elektor.es/110265
[5] http://beagleboard.org
[6] http://software-dl.ti.com/dsps/dsps_public_sw/sdo_tii/TI_Android_DevKit/02_02_00/index_FDS.html
[7] https://dl-ssl.google.com/android/repository/usb_driver_r03-windows.zip

Fotodiodo para medir rayos gamma

Detector de radiación con el BPW34

Burkhard Kainka (Alemania)

Cuando se trata de medir radioactividad, lo primero en lo que pensamos es en un contador Geiger-Müller. Pero los tubos contadores son difíciles de conseguir, aparte de caros, y en caso de hacernos con uno, todavía necesitaríamos tensiones de alimentación de algunos cientos de V. Sin embargo, es menos sabido que un fotodiodo normal y corriente como el BPW34 también pueda detectar los rayos X y gamma.

La radiación ionizante es potencialmente peligrosa y debemos mantenernos alejados la mayor distancia posible. Un contador Geiger de tubo pequeño no siempre basta para detectar un posible peligro. Al igual, el sensor semiconductor aquí descrito tiene una sensibilidad relativamente baja y sólo detecta las fuentes de radiación intensas. No obstante, basta para llevar a cabo interesantes medidas y experimentos.
La ventaja del fotodiodo es su reducida área de sensibilidad. El offset debido a los rayos cósmicos es muy bajo, y se distinguen señales de pequeñas fuentes entre el entorno mejor que con un tubo contador.

Radiación

Hablando de protección contra la radioactividad, lo más crítico son los rayos gamma. Penetran incluso gruesas paredes y resulta muy difícil lograr un apantallamiento efectivo. Continuamente, potentes rayos gamma provenientes del espacio atraviesan la atmósfera y las paredes no pueden hacerles frente. Los rayos alfa, por el contrario, tienen un rango de actuación muy reducido, normalmente ni siquiera atraviesan una hoja de papel. A esto se debe que la mayoría de contadores no puedan detectarlos. En tal caso, sólo es posible si el tubo contador tiene una fina apertura de mica. Los rayos beta alcanzan un mayor rango e incluso pueden atravesar chapas finas. La mayoría de tubos medidores captan principalmente los rayos gamma, así como la radiación beta con ciertas limitaciones.

Un diodo como detector

El comportamiento de un fotodiodo PIN BPW34 es similar al de los caros contadores. Las partículas alfa son atrapadas por el encapsulado de plástico del diodo. Sin embargo, los cuantos gamma no tienen problema en pasar y generan multitud de pares de huecos de electrones en la capa de barrera. Si polarizamos el diodo en sentido opuesto, absorberemos prácticamente todos los portadores de carga. El resultado es una pequeña corriente, que puede ser amplificada y medida. Las partículas beta también pueden generar una señal si tienen suficiente energía para atravesar la capa de barrera.
No obstante, las amplitudes de señal suministradas por el fotodiodo siempre son menores que las de un tubo contador, y por ello es necesario un amplificador de instrumentación con muy poco ruido.
Otro requerimiento para poder utilizar el fotodiodo como detector es garantizar su completo aislamiento de la luz, ya que en caso contrario el flujo de fotones eclipsaría a los resultados de la medida. En nuestro montaje hemos utilizado papel de aluminio.
Los diodos PIN, al contrario que los PN, disponen de una zona dopada N adicional. Esta capa conductora de alta impedancia (capa intrínseca, también llamada capa I) se encuentra entre las capas N y P. Como resultado obtenemos una capa aislante más profunda. Los fotones se encuentran con un mayor volumen al interactuar con el semiconductor. En los fotodiodos este concepto se aprovecha para conseguir el mayor número posible de portadores de carga por fotón, optimizando así la sensibilidad.

Otra opción a la hora de aumentar la sensibilidad consiste en utilizar una mayor superficie del sensor. Sin embargo, esto también tiene el inconveniente de aumentar la capacidad y que cada señal individual tenga menor amplitud en la tensión. Los semiconductores detectores de radioactividad disponibles comercialmente tienen una gran superficie y una capa I bastante gruesa. En comparación, los simples fotodiodos PIN como el BPW34 resultan menos sensibles, pero también mucho más asequibles.

Los fotodiodos PIN BPW34 y BPX61 son prácticamente idénticos, pero su encapsulado es distinto. Mientras que el (económico) BPW34 es de plástico, el BPX61 incorpora un encapsulado TO5 metálico con una ventana de cristal. Puede retirarse esta ventana cuidadosamente, dejando el chip al aire libre. Así, el diodo también será sensible a la radiación alfa.

Los rayos o partículas deben atravesar primero un papel de aluminio de 15 µm de grosor (el más común en este tipo de papel). Para la radiación gamma y beta esto es muy sencillo. Las partículas alfa con una energía superior a 4 MeV también son capaces de ello. La radiación atravesando el plástico del encapsulado en el fotodiodo genera pequeños flashes de luz que pueden ser detectados. Por ello, no es de extrañar que el BPW34 también sea sensible parcialmente la radiación alfa.

En principio todo semiconductor es sensible a la radiación ionizante. Lo increíble no es que un fotodiodo pueda detectar tal radiación, sino que la mayoría de las veces esta pasa desapercibida. Es bien conocido el efecto sobre las RAMs dinámicas, en las cuales la radiación puede modificar su contenido espontáneamente. También está el problema a la hora de diseñar productos electrónicos lo suficientemente inmunes, dada la elevada radiación cósmica. Los microcontroladores antiguos con grandes estructuras de transistores resultaban menos sensibles. Actualmente estas estructuras se hacen cada vez más pequeñas, y en ciertas circunstancias una partícula individual con alta energía puede provocar una perturbación.

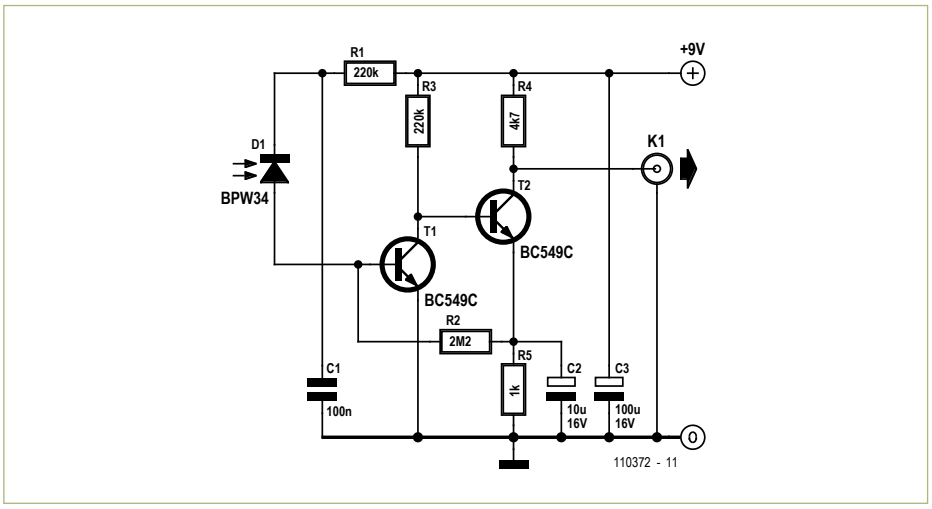


Figura 1. Amplificador de medida.

Amplificador

En algunos libros podemos ver cómo la etapa de entrada de los amplificadores de carga generalmente integra un operacional de bajo ruido con entradas FET. No obstante, aquí hemos ido por otro camino. La **figura 1** muestra el circuito del amplificador del sensor. Dos transistores bastan para amplificar la señal del fotodiodo. El amplificador acoplado directamente fija el punto de trabajo medio automáticamente, y los transistores NPN BC549C de bajo ruido se encargan de conseguir una buena relación señal-ruido.

La entrada del transistor en el amplificador en comparación es de baja impedancia, lo cual ofrece una buena reducción del ruido. La primera etapa utiliza su capacidad base-colector a modo de integrador, transformando los breves pulsos del fotodiodo en otros más largos, resultando así más fáciles de amplificar.

Para elevar la sensibilidad también podemos incrementar la tensión inversa en el diodo. Con ello se reduce la capacidad del mismo, aumentando al tiempo el grosor de la capa de bloqueo (o aislante). Éste permite hasta 32 V, pero lo óptimo es menos. A 9 V el diodo ya funciona bastante bien. Podemos también conectar varios fotodiodos en paralelo y alcanzar la sensibilidad que nos ofrece un pequeño tubo contador (por ejemplo el ZP1310).

Figura 2. Montaje de prueba del amplificador de instrumentación.

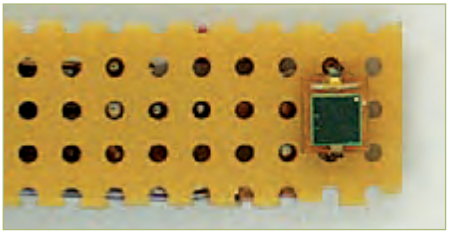


Figura 3. Sensor en la cara posterior.

Pintura luminiscente de radio

Un viejo reloj con los números luminosos es ideal para poner a prueba medidores de radiación. También puede tratarse de un despertador o brújula sacados de un mercadillo.

La pintura radiactiva se utiliza desde aproximadamente 1965 y va perdiendo poco a poco su intensidad luminosa. Si no sabe con certeza si su reloj es radiactivo, podemos hacer una sencilla prueba sin necesidad de electrónica alguna. Ante todo necesitamos una buena lupa. Apagamos las luces por la noche y dejamos que nuestros ojos se acostumbren a la oscuridad. Ahora observamos los indicadores luminosos con la lupa. Si su pintura contiene partículas radiactivas, podremos ver un ligero resplandor y parpadeo. A la vista de nuestros ojos, por así decirlo, se trata de procesos de desintegración individuales. Las partículas alfa resultantes excitan la pintura fluorescente. Si no vemos luz alguna, o se trata de una luz completamente uniforme, es seguro que no se trata de una muestra radiactiva. La pintura radiactiva se comporta de este modo ahora debido posiblemente a su antigüedad. En su origen lo más seguro es que se dieran tantos procesos a la vez, que resultase imposible distinguirlos individualmente.

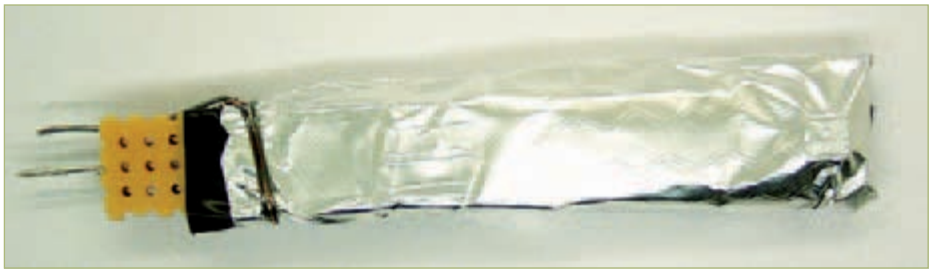


Figura 4. Todo perfectamente envuelto en papel de aluminio.

En la salida del circuito puede utilizarse un osciloscopio para analizar los resultados. Quien quiera escuchar los crujidos "característicos" de un contador Geiger en un altavoz, en el cuadro "cómo hacer audible la radiación" encontrará la solución adecuada.

Montaje

El circuito se ha montado en una tarjeta perforada (**figura 2**), y el fotodiodo se encuentra en la cara posterior de ésta (**figura 3**). Para mantener el circuito aislado de la luz se ha envuelto en papel de aluminio (**figura 4**). Hemos de utilizar el papel de aluminio especificado, lo suficientemente fino como para permitir que lo atraviesen las partículas beta. Al mismo tiempo el papel actúa como apantallado eléctrico.

Para que no haya cortocircuitos en el papel de aluminio, primero hemos de recubrir la tarjeta con cinta aislante, dejando al aire la ventana del fotodiodo. Sólo así posteriormente la envolveremos en papel de aluminio. No hemos de olvidar conectar el papel a masa.

Objetivos y resultados

Para el análisis lo mejor es utilizar un osciloscopio digital. Lo configuramos en modo AC. Un ajuste adecuado sería 50 mV/división y 0,2 ms/división. Algunos equipos permiten el modo persistente, con el cual se acumulan los resultados en la pantalla. No obstante, también podemos utilizar un osciloscopio analógico.

En estado de espera tendremos una banda de ruido amplificada de unos 30 mV_{SS} de amplitud (**figura 5**). Cuando se mide un cuanto gamma aparece un pulso positivo con sobrepaso negativo mucho menor. Si vemos señales con la misma amplitud en el área negativa, esto indica que el circuito no está apantallado de forma óptima

Cómo hacer audible la radiación

Los "crujidos" de un contador Geiger tienen algo especial. Pero por ahora, nuestro sensor diodo está mudo. Esto no tiene por qué seguir así. Basta con conectar un comparador para alargar los impulsos ligeramente, y así podrán sonar crujidos en el altavoz. El circuito probado aquí utiliza un comparador LM311, que proporciona un pulso de salida cuando los impulsos a la entrada superan el valor de umbral fijado por el potenciómetro. El transistor a la entrada alarga los impulsos, de modo que puedan oírse posteriormente. Posteriormente, la señal de salida controla por ejemplo unos auriculares, un amplificador de audio con altavoz incorporado o un pequeño altavoz activo individual.

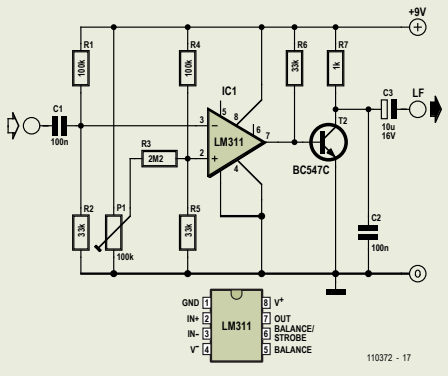


Figura 5. Estado de espera.

Figura 6. Medida durante 30 segundos de un reloj de bolsillo con indicadores luminosos.

Figura 7. Medida durante 30 segundos de un mineral que contiene uranio.

y está reaccionando a impulsos HF. Por el contrario, la radiación en la que queremos centrarnos genera pulsos positivos. En la **figura 6** pueden verse las señales acumuladas durante 30 s producidas por un antiguo reloj con indicadores luminosos de radio.

La **figura 7** muestra la medida con otra muestra radiactiva. Se trata de un trozo de uranitita (o "pechblenda"), el mineral presente en la naturaleza del que se extrae el uranio. Esta vez también se midió durante 30 segundos. Puede reconocerse fácilmente que esta muestra tiene una mayor actividad. Aparte, puede verse cómo tiene una distribución energética distinta. Hay muchos impulsos de más de 100 mV que en la representación no aparecen. Al contrario que con un tubo contador Geiger, con el detector semiconductor puede averiguarse la energía de las distintas partículas. Podemos obtener conclusiones sobre qué el tipo de núcleos que se están desintegrando en cada momento. En el caso de la uranitita se trata de procesos de desintegración del uranio. En cuanto al color de la luz, se debe probablemente a los núcleos del radio.

Gracias a la posibilidad de acumular mediciones durante un largo periodo de tiempo podemos analizar situaciones en las que cabe esperar una radiactividad nula o muy reducida. En este caso funciona mejor que con un tubo contador, pues prácticamente no existe un nivel de cero. En un tubo contador siempre hay algunos impulsos procedentes de la radiación cósmica de fondo. Ciertamente, estos potentes rayos gamma impactan también contra el fotodiodo, pero la superficie es tan reducida que resulta extremadamente raro. Por ello, las señales útiles destacan claramente en el estado de espera. La **figura 8** muestra una medida con un trozo de galena, un mineral que no debería (casi) irradiar. Sin embargo, tras media hora se obtienen dos picos bien visibles. Obtenemos un resultado similar con una piedra de granito, famosa por su baja radiactividad.

Una fuente radiactiva también puede tratarse de un dispositivo o componente tecnológico que no fue diseñado cumpliendo con los estándares actuales. Es bien conocido por ejemplo el caso de las lámparas de neón y sus estabilizadores de tensión que funcionan por debajo de los 100 V con impurezas radiactivas. El autor sospechaba de un viejo tubo luminoso ruso de 75 V/3 mA (**figura 9**) que tenía desde hace bastante tiempo. En la cubierta exterior puede verse una pequeña carcasa de metal, bajo la cual se identifica fácilmente una extraña especie de cápsula. Debajo hay

Productos de la desintegración del radón

Si no tenemos a mano ninguna otra sustancia radiactiva para testear, podemos obtenerla del medio. Estamos continuamente inmersos en un entorno radiactivo. De la superficie de la tierra siempre escapa algo de radón. Este gas tiene un tiempo de desintegración apreciable, aparte de generar otros núcleos radiactivos tras este proceso. Podemos obtenerlo fácilmente por nuestra cuenta.

Dentro de casa, nos hacemos con un trozo de cable de cobre lacado (de 0,2 mm de calibre) y lo estiramos debidamente. Aplicamos en el cable una alta tensión negativa de entre -5 kV y -10 kV. Al pasar diez minutos (¡previa desconexión de la alta tensión!) lo frotamos con una tira de papel. En éste encontraremos unas trazas oscuras: se trata de polvo atraído por la carga en el cable. En estas partículas podemos encontrar muchos productos de la desintegración del radón.

El motivo es obvio: en la desintegración los núcleos nuevos alcanzan una gran velocidad. Toman energía de una de las capas de electrones y se cargan positivamente. Por ello, son atraídas por el cable cargado negativamente.

Si ahora acercamos este papel "sucio" a un medidor de radiactividad, descubriremos una elevada actividad. No resulta peligroso recoger los isótopos, sino inhalarlos. De esta manera podemos analizar la cantidad de radón que hay en distintas habitaciones. En el sótano hay considerablemente más, pues el radón viene de abajo.

Se describe un generador de alta tensión apto por ejemplo en el "miniproyecto" del ionizador de la edición de Elektor 3/2009, pág. 66 (www.elektor.es/071072), en el cual se conectan reiteradamente en cascada dos etapas (dos condensadores y dos diodos) hasta lograr 5 kV.

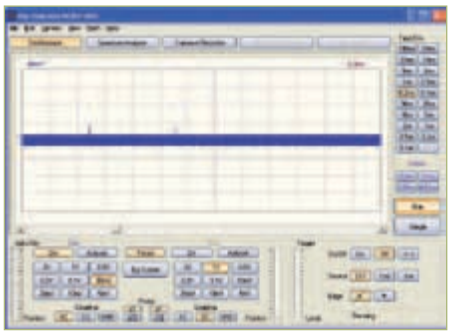


Figura 8. Medida durante 30 minutos de galena.

Figura 9. Tubo luminoso radiactivo.

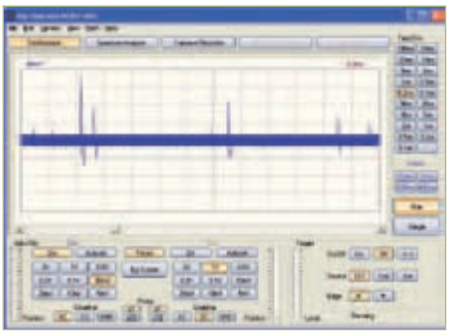


Figura 10. Medida del tubo luminoso de la figura 9.

un pequeño agujero. La medida prolongada durante media hora revela la existencia de impulsos con una energía especialmente alta (**figura 10**). Y eso a pesar de que la radiación tiene que atravesar también el cristal del tubo.

Conclusión

En este artículo hemos presentado el sensor y un amplificador de instrumentación realmente simple. Si colocamos el circuito junto con el comparador descrito en el cuadro de texto y un altavoz en una carcasa, obtendremos un dispositivo para ser utilizado en el exterior, por ejemplo para analizar minerales en una cantera. Si combinamos el comparador con un contador digital, también podremos conocer la actividad total. Con un circuito adicional de Muestreo y Retención (Sample & Hold) pueden registrarse los niveles de energía. Posteriormente, los resultados pueden representarse a modo de espectrograma de energía.
También sería interesante analizar otras muestras durante un tiempo prolongado. Por ejemplo, el cloruro de potasio es un emisor leve de radiación beta. Sería interesante comprobar si el fotodiodo es capaz de demostrarlo.

(110372)

Bibliografía y enlaces:

- Videos del detector BPW34: http://www.youtube.com/user/bkelektronik
- **Nota de aplicación Maxim 2236, detector de radiación gamma y fotónica** http://pdfserv.maxim-ic.com/en/an/AN2236.pdf
- **Erhan Emirhan y Cenap S. Özben: PIN Fotodiode Based X and γ Ray Detectors** http://thm.ankara.edu.tr/tac/YAZOKULU/yazokulu6/dersler/06-09-2010/erhan-emirhan-cenap-ozben-pin-fotodiode.pdf
- **C. W. Thiel: An Introduction to Semiconductor Radiation Detectors** www.physics.montana.edu/students/thiel/docs/Detector.pdf
- **Elektor 1/1980 S. 48-53: Contador Geiger-Müller** www.elektor.com/Geiger-Müller-Zähler
- Funkschau 1986, edición 21, págs. 63-69: medidor de radiactividad en miniatura
- Funkschau 1986, edición 21, págs. 99-103: Cómo medir la radiactividad
- Thomas Rapp: experimentos con un contador Geiger casero, cámara de chispas y niebla, editorial Franzis, 2008

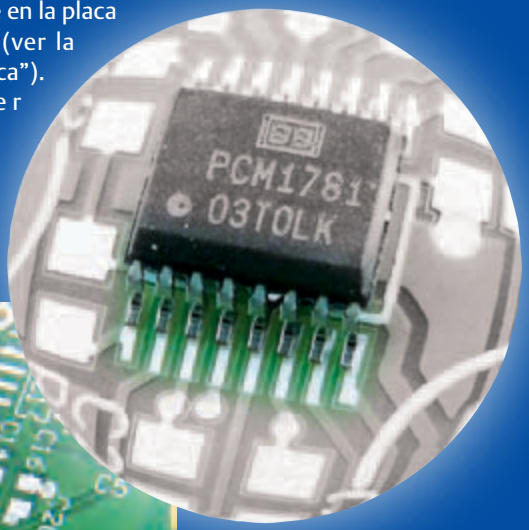
Diseño de placas:
¡Ojo con los integrados pequeños!

Thijs Beckers (Redacción Holanda)

Quizás hayas leído los artículos sobre nuestro sistema DSP en el número anterior y en esta edición (y si no ¡en serio que deberías hacerlo!). En las dos primeras partes describimos las posibilidades de un DSP y del software de programación a utilizar. En la tercera parte, que se publicará en la edición de septiembre, presentaremos el esquema y trataremos sobre la placa impresa, entre otras cosas.

En esta placa se hallan diversos integrados SMD con varios encapsulados diferentes. Así se encuentran los operacionales utilizados en un encapsulado SOIC (Small-Outline Integrated Circuit) con una distancia entre terminales de 1,27 mm, el integrado DSP que cuenta con un encapsulado TQFP (Thin Quad Flat Pack) con 52 terminales que distan 0,65 mm entre sí, al igual que los terminales del convertidor A/D, que se encuentran en un encapsulado TSSOP (Thin Shrink Small-Outline Package). En el caso del convertidor de velocidad de muestreo, tenemos que dar un paso aún más pequeño con un encapsulado TQFP y un "pitch" – la terminología inglesa de la distancia entre terminales – de 0,50 mm. También por eso decidimos suministrar el módulo completamente construido; muchos tendrán dificultades al montar en la placa este tipo de componentes pequeños (por desgracia no se venden en tamaños mayores).

Una pequeña "trampa" en el diseño de la placa fue el encapsulado del convertidor D/A. Este integrado de Texas Instruments viene con un encapsulado SSOP/QSOP (Shrink Small-Outline Package/Quarter-Size Small-Outline Package) que se parece mucho al encapsulado TSSOP del convertidor A/D, pero en el que la distancia entre los terminales alcanza 0,635 mm. ¡Una diferencia de tan sólo 0,15 mm ("15 centésimas de un milímetro")! Esta diferencia no es apreciable en la placa a simple vista (ver la "foto panorámica"). Para poder demostrar esta mínima diferencia hemos colocado el integrado convertidor D/A tanto en su sitio propio (a la izquierda, con un pitch de 0,635 mm) como en el sitio del convertidor A/D (a la derecha, con un pitch de 0,65 mm).

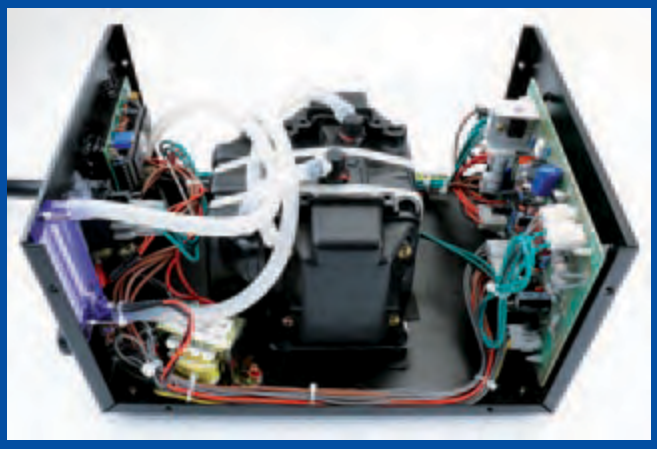


Cuando ampliamos mucho, observamos que los terminales coinciden perfectamente con los islotes si el integrado se encuentra en su propio sitio (ver ampliación en la parte inferior izquierda). Pero si lo colocamos en el sitio del convertidor A/D (ver ampliación en la parte superior derecha), observamos que hay una diferencia de más o menos la mitad de la anchura de un terminal y que no se puede soldar bien el integrado en este sitio. Quizás se pueda hacer, pero no queda muy bonito y la probabilidad de cortocircuitar aumenta bastante.

Por supuesto, un diseñador SMD experto se encogerá sus hombros y dirá "¡¿Esto lo sabe todo el mundo?! Siempre hay que mirar la distancia entre los terminales y no solo el nombre del encapsulado", pero un hombre prevenido, vale por dos (¡ahora ya lo sabes!) y tengo que admitir que no sabía que existía también una variante de encapsulado con una distancia de terminales de 0,635 mm. Bueno, vamos a decir que para aprender nunca es tarde...

(110394)

Huracán sobre los SMD



Thijs Beckers (Redacción NL)

Nuestro compañero Luc Lemmers ha cambiado de parecer. Se ha dejado convencer para adquirir una estación de refusión de aire caliente para casa. Hace ya un tiempo que trabajamos satisfactoriamente con el Aoyue 852A+ en el laboratorio. Tengo que decir, y hablo por experiencia propia, que para la soldadura (y desoldadura) de SMD es realmente mucho más fácil, agradable y además más limpio que con una estación de soldadura 'antigua'. Por otra parte, es casi imposible desoldar integrados SMD con un soldador 'normal' sin accesorios especiales. Con una estación de refusión de aire caliente es cuestión de mover la tobera por los terminales del integrado hasta que el estaño se funda. Y esto se hace en 5 segundos, si se hace bien.

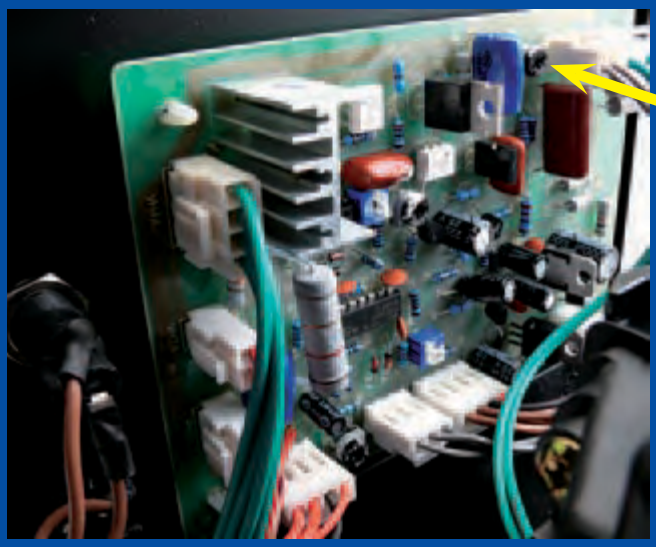
Pero bueno. La elección de Luc recayó sobre el 852 de Aoyue (sin 'A+'). Una estación perfecta por no mucho dinero. No hay nada en contra hasta que diriges la tobera hacia una resistencia pequeña... un huracán de la categoría 5 se queda en nada si lo comparas con el flujo de aire saliente de la tobera. Adiós resistencia.

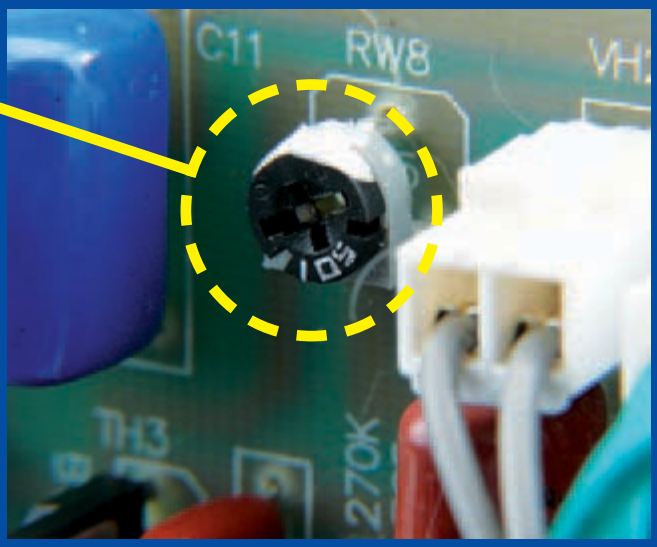
'¿Bajamos el flujo entonces?' Sí, si se pudiera... porque el potenciómetro de la regulación del flujo de aire ya está en la posición mínima.

No, esto no es que fuese 'sacarlo de la caja y trabajar directamente con él'. A Luc como electrónico le parecía una tontería devolver el aparato sin mirar primero si podía resolverlo él mismo. ¡A coger el desatornillador y abrir la caja!

El contenido era mejor de lo que esperábamos. Por el precio que pagas, compras un aparato con un acabado bastante decente. Una placa ordenada, un compresor adecuado y un display estándar. Un tiquismiquis se daría cuenta de que el transformador no lleva marca CE...

En la placa pudimos ver claramente que había un potenciómetro de ajuste (ver foto) en serie con el potenciómetro de la configuración del aire caliente. ¿Sería suficiente si le girábamos un poco? Sí, Luc tuvo suerte. Con el potenciómetro de ajuste se pudo limitar aún más el flujo de aire que utilizando solamente el potenciómetro de la parte frontal. Después de experimentar un poquito encontramos una configuración mejor y pudimos cerrar la caja de nuevo. Después intercambiamos los botones giratorios de la parte frontal, ya que los ajustes del flujo de aire y





LABCENTER

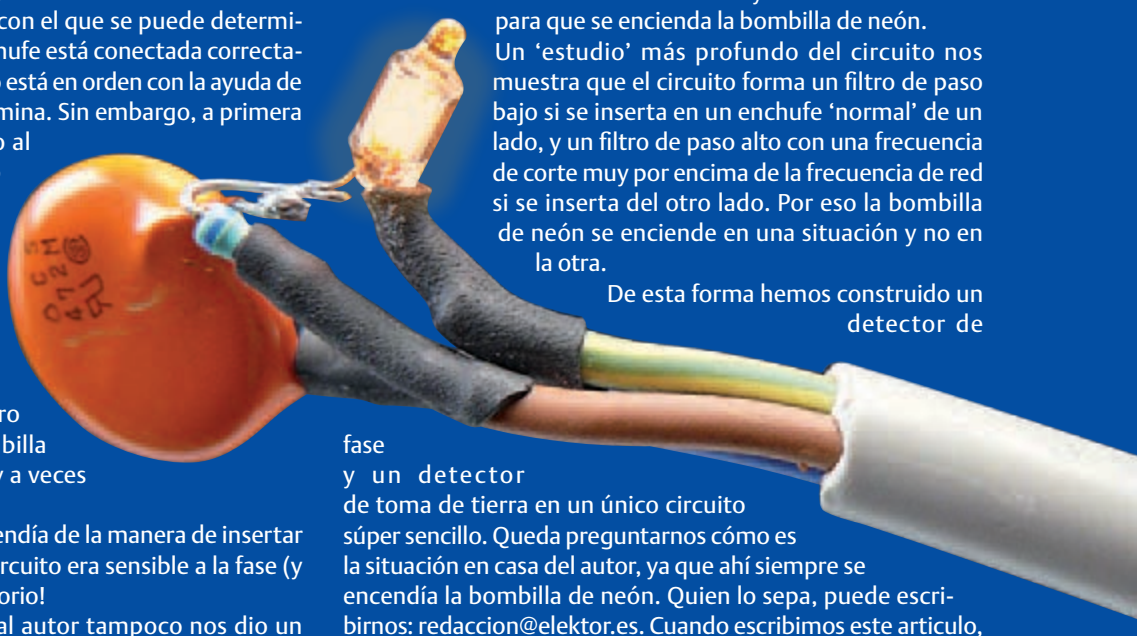
la temperatura se leen casi siempre del display, y no de los botones giratorios. Así que es más lógico intercambiar la ubicación de los botones giratorios. Por supuesto que la indicación de la escala ya no es correcta, pero esto no supone ningún problema si la estación es para uso propio.
Con una sensación de satisfacción tenemos ahora una estación de refusión realmente lista para su uso, sin que tener que esperar varios días de haberlo hecho todo de forma 'oficial' (devolverlo). Y luego esperar a que lo devuelvan y que funcione correctamente.
¡Ten en cuenta que probablemente, después de estas manipulaciones, se pierde la garantía! No es que a Luc le importe mucho, porque si tiene algún problema, ¡simplemente lo arreglará él mismo!
Ojo, ¡No empieces a girar el potenciómetro de ajuste equivocado! Aparte de los que hemos tratado aquí, hay varios más. Sirven, entre otras cosas, para la regulación de la temperatura. Esto da que pensar: ¿Alguien sabe cómo se calibra la temperatura de la estación de refusión? Por ejemplo, ¿Dónde se mide la temperatura para el aire caliente?
Y por supuesto: Ojo con el ajuste del flujo del aire; ¡el aparato lleva tensión de red!
Podéis enviarnos vuestras opiniones a redaccion@elektor.es

(110261)

Puesta a tierra

Thijs Beckers (Redacción Holanda)

Nuestro compañero más joven Raymond Vermeulen, estaba trabajando en un circuito de un autor valón para nuestra edición especial de verano, para muchos también conocida como la 'guía de los semiconductores'. Raymond no es precisamente hábil en la lengua francesa, pero cualquier electrónico puede 'leer' los circuitos. Y este era muy sencillo.
Se trata de un circuito detector con el que se puede determinar si la toma de tierra de un enchufe está conectada correctamente. Este circuito indica si todo está en orden con la ayuda de una bombilla de neón que se ilumina. Sin embargo, a primera vista parecía pasar algo extraño al circuito tal y como el autor lo había presentado.
Para probar el funcionamiento del circuito Raymond lo 'dotó' de un cable IEC (ver foto). Una prueba en el laboratorio parecía confirmar su buen funcionamiento. Sin embargo, al insertar el conector en cualquier otro enchufe del laboratorio, la bombilla de neón se encendía a veces sí y a veces no.
Se dedujo rápidamente que dependía de la manera de insertar el conector. Aparentemente el circuito era sensible a la fase (y el neutro). ¡Excepto en el laboratorio!
Un correo electrónico enviado al autor tampoco nos dio un resultado definitivo. Según él, el circuito debería funcionar sin que importase cómo se insertara el conector en el enchufe...
Ahora sí sabemos por qué el circuito funciona en nuestro laboratorio si se inserta el enchufe de ambos lados. Es que el laboratorio se provee de tensión de red a través de un transformador de aislamiento. Esto hace que los diseñadores puedan trabajar de forma más segura. Sin embargo, el terminal de tierra de los enchufes sí está conectado directamente a la toma de tierra. De modo que podrías decir que ambos conductores de la red flotan. No hay realmente una conexión 'fase' o 'neutra', solamente una tensión alterna flotando. La conexión capacitiva entre ambos conductores y la tierra es suficiente para que se encienda la bombilla de neón.
Un 'estudio' más profundo del circuito nos muestra que el circuito forma un filtro de paso bajo si se inserta en un enchufe 'normal' de un lado, y un filtro de paso alto con una frecuencia de corte muy por encima de la frecuencia de red si se inserta del otro lado. Por eso la bombilla de neón se enciende en una situación y no en la otra.
De esta forma hemos construido un detector de fase y un detector de toma de tierra en un único circuito súper sencillo. Queda preguntarnos cómo es la situación en casa del autor, ya que ahí siempre se encendía la bombilla de neón. Quien lo sepa, puede escribirnos: redaccion@elektor.es. Cuando escribimos este articulo, aún no lo sabíamos.
Queda pendiente el esquema del circuito (estate atento a la próxima edición especial de verano). El electrónico de verdad ya se habrá formado una idea de cómo es el circuito.



(110396)

Modificación de la placa de transporte del ElektorWheelie

Jan Visser (Laboratorio de Elektor)

Como colaborador del laboratorio, entro a menudo en contacto con los problemas técnicos que sufren los lectores y colegas en sus proyectos. En este artículo del Labcenter, me gustaría resaltar uno relacionado con el ElektorWheelie (ver la edición de Elektor de julio/agosto y de septiembre de 2009 y/o [1]).
Después de un uso intensivo del ElektorWheelie, resulta que los tornillos "wheels square" (ver **foto de la izquierda**) se pueden deformar o incluso romper debido a un uso intensivo o una excesiva oscilación. Para evitarlo hemos inventado la siguiente modificación.

Coge cuatro tubitos de latón o aluminio con un diámetro interno de 5 mm y un diámetro externo de 6 mm. Los tubitos se pueden obtener cortándolos de un tubo más largo, el cual tiene que tener una longitud de 25 mm. Estos caben perfectamente en los huecos de las ruedas de plástico del ElektorWheelie. Puedes encontrar los tubitos de latón o de aluminio que se necesitan para esto en varias tiendas especializadas de modelismo o también en ferreterías. También en Conrad [2] puedes adquirir un tubo adecuado. Busca con el número de artículo 297321.
La idea es pegar los tubitos de latón o de aluminio en estos huecos (ver **foto de la derecha**). Utiliza para eso un pegamento de contacto y aplícalo sobre la parte exterior de los tubitos. Utiliza la placa de transporte como molde, de modo que los tubitos queden en el sitio correcto de la rueda y coloca todo el conjunto con los tubitos en los huecos de la rueda.

Asegúrate de que los tornillos y la placa de transporte no se peguen con la rueda. Más tarde la tenemos que poder retirar sin ningún problema.
Deja que el pegamento se seque el tiempo suficiente (una noche) y retira completamente la placa de transporte con los tornillos de la rueda. Entonces los tubitos quedan muy bien fijados a la rueda y en el sitio correcto. Para que el conjunto sea aún más fuerte, rellenamos con pegamento los espacios vacios de los huecos del plástico. Para eso puedes utilizar tanto pegamento de contacto como pegamento térmico.
En el momento en que el pegamento se haya secado por completo, podemos volver a montarlo todo en el ElektorWheelie. No te olvides de montar de nuevo la tuerca central (con la que la placa de transporte está fijada sobre el eje), con un poco de Loctite, para evitar que se suelte.
Con esta sencilla modificación, reduces las fuerzas que tienen que soportar los tornillos "wheels square" y los golpes que tienen que aguantar durante las oscilaciones. Así la rueda gira directamente y se evita que los tornillos se deformen o incluso se rompan.

(110395)

Enlaces Web:
[1] www.elektor.es/wheelie
[2] www.conrad.com

LABCENTER

E-blocks: Flowcode RC5

Añada un control remoto a sus proyectos

RC5 es un protocolo de comunicaciones para un mando a distancia de infrarrojos. Es de uso común en muchas aplicaciones domésticas, y los dispositivos compatibles RC5 son tan baratos como los circuitos integrados. En este artículo vamos a ver cómo podemos aprovechar el RC5 y añadir un mando a distancia transceptor de infrarrojos a nuestros proyectos electrónicos, usando E-blocks.

Sean King (Reino Unido)

RC5, como un estándar de mandos a distancia por infrarrojos para televisores y aplicaciones para el consumidor, ha estado a nuestro alrededor durante tanto tiempo que se ha convertido en un estándar no oficial para muchas aplicaciones. Pero ¿cómo funciona el RC5? Para responder a esta pregunta tenemos que mirar en el circuito de recepción y en el programa requeridos por el microcontrolador que va a recibir y descodificar las instrucciones RC5. Primero vamos a ver, con algo más de detalle, el protocolo RC5, que nos dará una idea más clara del circuito y programa requeridos para generar y descodificar las señales.

Protocolo

RC5 es un estándar de transmisión de datos relativamente lento. Cada bit tiene un período típico de 1778 µs (con grandes variaciones toleradas). Una transmisión RC5 normal contiene 14 bits de datos (casi 25 milisegundos por mensaje). En la **Figura 1** se muestra la descomposición de un mensaje RC5 típico, donde puede verse que hay 2 bits de inicio + 1 bits de conmutación; + 5 bits de direcciones + 6 bits de comandos. La función de cada uno de estos bits se explica más abajo.

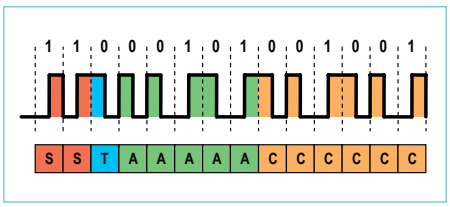


Figura 1. Estructura de un mensaje RC5. Inicio = 1 + 1. Conmutación = 0. Dirección = 5 (valor típico VCR). Comando = 9.

Bits de inicio: son dos bits que, cuando ambos tienen el valor '1', indican el principio de un mensaje y pueden ser usados como referencias de tiempo para los datos siguientes.

Bit de conmutación: cambia de estado cada vez que se pulsa una nueva tecla en el mando. Mensajes repetidos con el mismo valor de bit de conmutación indican que el botón del mando sigue pulsado y que los mensajes están siendo transmitidos en modo auto-repetición (típico del funcionamiento del control de volumen).

Bits de dirección: se utilizan cinco bits de dirección para identificar el destinatario deseado del mensaje. Algunas direcciones han sido asignadas a dispositivos comunes.

Productos y Servicios de Elektor:

- Placa E-blocks RC5 de Infrarrojos (EB060)
- E-blocks Multiprogramador dsPIC/PIC24 (EB064)
- E-blocks pantalla LCD gráfica (EB058)
- Placa E-blocks de conmutación (EB007)

Precios y detalles de cómo hacer el pedido en www.elektor.es/e-blocks

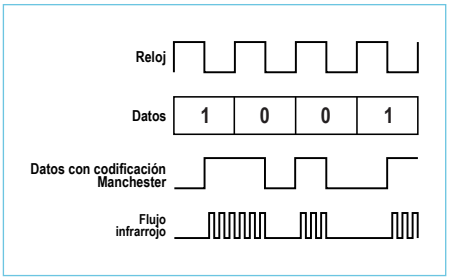


Figura 2. Esquema de codificación RC5.

TV principal	= 0
Grabador de vídeo	= 5
Receptor de satélite	= 8

Bits de comando: los seis bits de comandos indican la operación a realizar. Las teclas numéricas en un mando a distancia son, por lo general, transmitidas como su valor numérico. El resto de funciones puede variar, pero podemos encontrar tablas de valores estándar.

Codificación

Considerando el canal de comunicación, el transmisor tiene un trabajo mucho más fácil que el receptor. El transmisor impone la temporización de la señal y la potencia, mientras el receptor debe adaptarse para corresponder a las características de cualquier transmisor compatible y los efectos del medio de transmisión entre ambos. Un problema fundamental con muchos sistemas de comunicación es que el receptor tiene que saber cuándo debe comenzar a 'muestrear' la señal de entrada (es decir, tiene que regenerar un reloj para la trama de datos). Para solventar este problema, RC5 usa lo que se conoce como 'codificación Manchester'. En la 'codificación Manchester' cada bit es representado por la dirección de una transición en mitad de su ranura de tiempo (ver **Figura 2**). Esto garantiza que habrá, al menos, una transición asociada con cada bit, ayudando a mantener la sincronización de tiempos en el receptor (aunque a expensas de un incremento del ancho de banda). Los valores de los bits precedentes y sucesivos determinan si también ha habido una transición al principio o al final de la ranura de tiempo del bit.

Modulación

El proceso final antes de la transmisión de una señal RC5 es el de aplicar una modulación de 36 kHz. Esto ayuda al receptor a eliminar las perturbaciones provocadas por la luz ambiente, establecer su control automático de ganancia e ignorar las señales transmitidas en otras frecuencias de modulación.

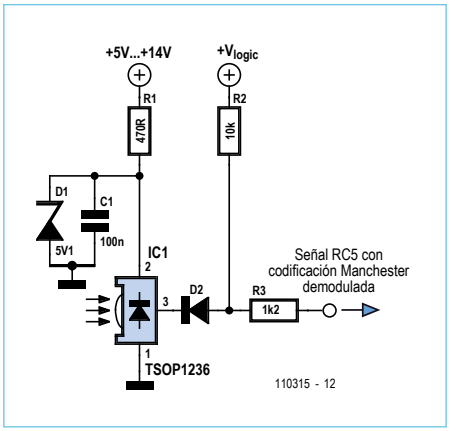


Figura 3. Circuito del receptor.

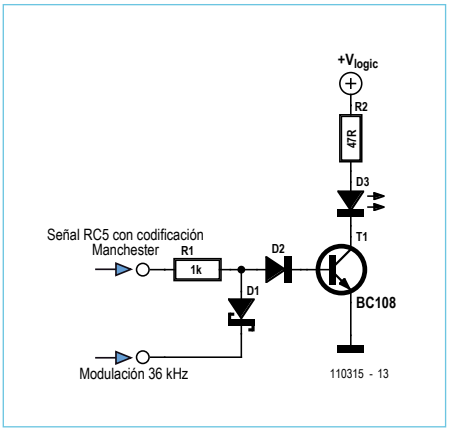


Figura 4. Circuito del transmisor.

El circuito para el RC5

Antes de examinar el programa, vamos a ver qué apariencia tiene el circuito. En la **Figura 3** podemos ver el circuito del receptor. El circuito es bastante sencillo: el receptor está formado por un módulo de receptor óptico (TSOP1236) que es sensible a la luz infrarroja. Este receptor es realmente una 'bestia' bastante complicada que incluye la circuitería para suprimir la señal de modulación de 36 kHz y que también maneja la complicada tarea del Control Automático de Ganancia (CAG) para la señal infrarroja. Pero, en lo que se refiere a nuestro circuito, se comporta como un transistor que se dispara por una señal óptica, en el que R1, D1 y C1 proporcionan una tensión de 5 V. La señal demodulada es llevada a la entrada de un microcontrolador. Esta línea es mantenida a nivel alto por R2. Cuando se recibe un pulso infrarrojo, el 'transistor' es activado, el cual cambia la salida a 0V. La **Figura 4** muestra el transmisor: cuando el microcontrolador envía un nivel lógico 1, T1 es activado y se envía el mencionado pulso (señalar que el microcontrolador tiene que generar los pulsos de 36 kHz, ya que usamos un diodo infrarrojo sencillo).

Sistema de desarrollo

Para el propósito de este proyecto, tanto el transmisor como el receptor están contenidos en una nueva placa de E-blocks: la placa de infrarrojos EB060. Es un diseño ordenado, que incluye un zócalo para un circuito integrado PIC10F independiente, que podemos usar para configurarlo como un modulador para el transmisor, liberando de trabajo a nuestro microcontrolador principal. En **la Figura 5** podemos ver dicha configuración. Aquí estamos usando una placa Multiprogramador dsPIC; una placa gráfica con LCD y una placa de conmutación que comparte el puerto A; y una placa de infrarrojos en el puerto E. El dsPIC que estamos usando es un dsPIC30F2011, un dispositivo de 18 terminales.

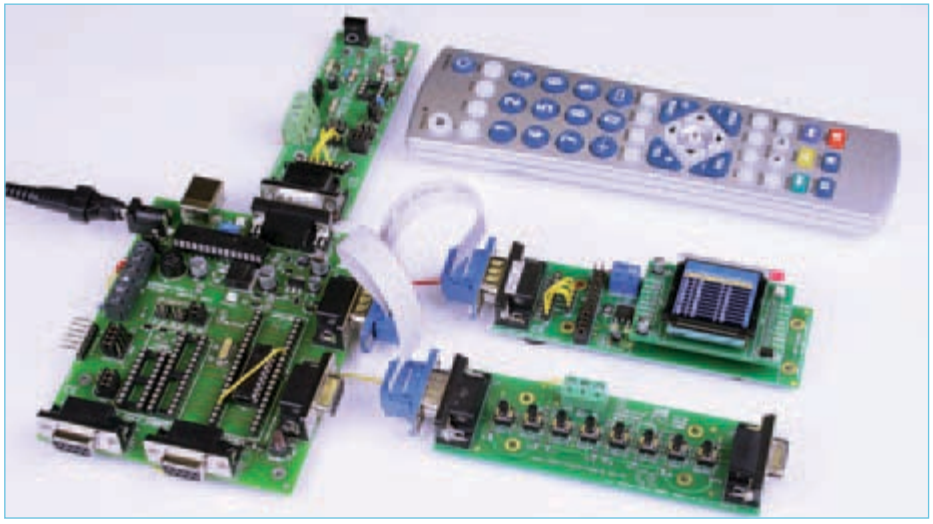


Figura 5. Configuración de prueba con los módulos de los E-blocks EB060, EB064, EB058 y EB007.

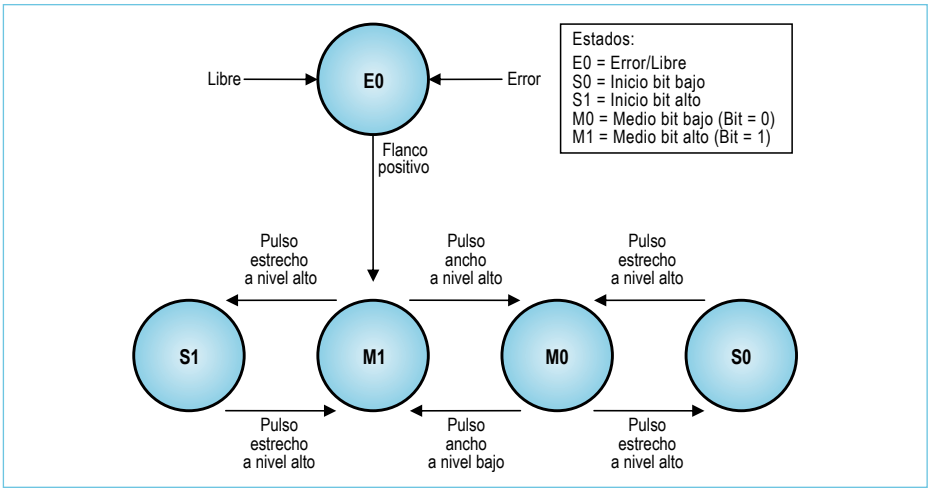


Figura 6. Máquina de estado del descodificador RC5: estados y transiciones.

Podríamos usar un PIC de 8 bits convencional, pero la placa dsPIC es más espectacular y nueva, por lo que nos hemos decidido por ella. Ahora que ya hemos explicado el circuito y el esquema de codificación, vamos a hablar del descodificador. Es aquí donde el verdadero trabajo de desarrollo de un sistema de este tipo comienza.

Descodificando la señal

El sistema RC5 ha sido diseñado para ser tolerante con las variaciones de temporización significativas. Esto era un requisito para los mandos a distancia previos, con osciladores poco precisos y baterías que trabajaban con una gran variedad de estados de carga. Los dos bits de inicio siempre están presentes en las transmisiones RC5 y pueden ser usados para sincronizar el temporizador del receptor al inicio de cada mensaje, permitiendo que cada bit de datos pueda ser muestreado en el momento adecuado. Esto es similar a la configuración "bit-banging" (emulador software de comunicación serie), usada a menudo en los programas de receptores RS232. Sin embargo, los pequeños errores en la medida del período de un bit se acumulan a lo largo de varios bits, que puede llevar a errores de detección en los bits al final del mensaje. Gracias a la transición garantizada en mitad de cada período de bit, un mensaje con 'codificación Manchester' puede ser considerado que está formado por una combinación de cuatro tipos de pulsos:

1. un pulso ancho a nivel alto;
2. un pulso estrecho a nivel alto;
3. un pulso ancho a nivel bajo;
4. un pulso estrecho a nivel bajo.

Estas definiciones pueden tener una gran tolerancia de temporización, aplicada a dichos pulsos, antes de que haya confusión entre un pulso corto y uno largo. Esto elimina la necesidad de conocer la frecuencia exacta de transmisión, evitando errores de acumulación con cada bit detectado, a la vez que permite la descodificación usando programas basados en una máquina de estado. Así pues, nuestra primera tarea es la de definir los tiempos que representan los pulsos 'anchos' y 'estrechos' que se derivan de los valores ideales del protocolo RC5, dentro de una tolerancia estándar de ±¼ bit de periodo (444 µs). Podemos ver esto en la **Tabla 1**.

Tabla 1. Tiempos de Ancho de pulso

	Min.	Ideal	Max.
Ancho (µs)	1334	1778	2222
Estrecho (µs)	444	889	1333

Descodificando la máquina de estado

En la descodificación de la señal necesitamos tener una técnica que podamos usar para capturar la trama de datos infrarroja y convertirla en el dato numérico sencillo que necesitamos. En la práctica usamos un truco de programa llamado 'máquina de estado', para hacer esto. Podemos ver cómo se representa esto esquemáticamente en la **Figura 6**. La máquina de estado comprende cinco estados. Las flechas entre los estados indican los caminos disponibles entre los estados y los eventos que provocarán que se produzca el cambio. Los estados M0 y M1 representan la recepción correcta de un bit de datos (0 y 1, respectivamente). Cada vez que la máquina de estado entra en uno de estos estados, el valor del bit correspondiente puede ser desplazado en el registro de desplazamiento de recepción.

Trampas de error

Si se produce un evento que no corresponde con un camino del estado actual, el sistema genera un error y vuelve al estado E0. Además de las transiciones a estados ilegales, el sistema volverá al estado E0 si el ancho del pulso medido es demasiado cor-

Figura 7. Pantalla de configuración de Flowcode RC5.

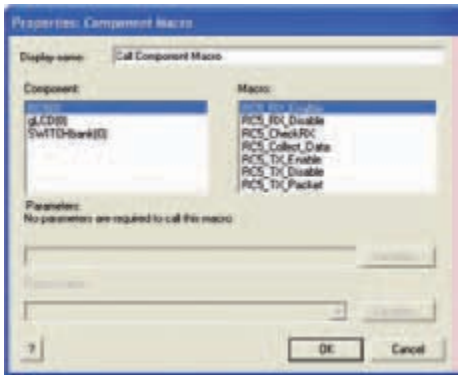


Figura 8. Pantalla de configuración de Flowcode RC5.

to o demasiado largo para representar una señal válida.

Ejemplo usando la Figura 6
El receptor está en reposo y la máquina de estado está en el estado E0. Se detecta un flanco positivo el cual cambia al sistema al estado M1 e introduce un 1 en el registro de desplazamiento del receptor. Un pulso estrecho, de nivel alto, cambia al sistema al estado S1. Un pulso estrecho, de nivel bajo, devuelve al sistema al estado M1 e introduce otro 1 en el registro de desplazamiento del receptor.
Esto completa la detección de los dos bits de inicio y debería ser siempre la secuencia inicial de un mensaje RC5.
Un pulso ancho, de nivel alto, cambia al sistema al estado M0 e introduce el Bit de Conmutación en el registro de desplazamiento del receptor con un valor 0. Los pulsos restantes cambian el sistema en torno a esta máquina de estado hasta que todos los bits de datos hayan sido recuperados. En la práctica, la máquina de estado se implementa estableciendo el estado de un cierto número de variables en un programa.

Realización con Flowcode para dsPIC/PIC24

El programa final fue implementado con Flowcode para dsPIC/PIC24, el cual incluye un nuevo componente 'macro' diseñado para trabajar con algunas variaciones del estándar RC5 que se usa comúnmente. El componente Flowcode RC5 permite una gran variedad de opciones de las señales RC5, tanto para transmisión como para la recepción, incluyendo la inversión de la señal recibida para compensar el circuito detector, así como un filtrado opcional basado en la dirección del mensaje de entrada (ver **Figuras 7** y **8**). Las macros proporcionadas permiten que se puedan crear los mensajes completos a partir de sus elementos individuales (Comando, Dirección, Conmutación) y que puedan transmitirse usando un único comando. También permiten detectar los mensajes recibidos y que los componentes individuales de un mensaje recibido puedan ser recuperados. Para ayudar en la depuración del programa, también usamos la placa gráfica de visualización de Bloques Electrónicos. Fue muy útil en la visualización de la secuencia de los comandos enviados por infrarrojos y por la gran cantidad de caracteres que nos permitía ver, entre ellos, la dirección y parte de los datos de la señal RC5, con lo que pudimos comprobar que nuestro algoritmo funcionaba correctamente.
Ha sido el trabajo inteligente de muchos ingenieros lo que nos ha permitido colocar nuestra TV en el tráfico directo de la sobrecarga infrarroja que proporciona la luz del sol y conseguir que nuestros mandos remotos funcionen correctamente por menos de lo que vale un puñado de componentes. Para solucionar este problema de los altos niveles de luz de fondo de la radiación infrarroja, los ingenieros establecieron que el estándar RC5 debía usar un tipo de modulación de pulso codificado. Los datos infrarrojos consisten en tramas de haces infrarrojas a una frecuencia de 36 kHz. Esta señal es modulada por el microcontrolador de manera que un 1 lógico pueda ser representado por la presencia de una trama infrarroja y un 0 lógico esté representado por la ausencia de cualquier señal infrarroja.

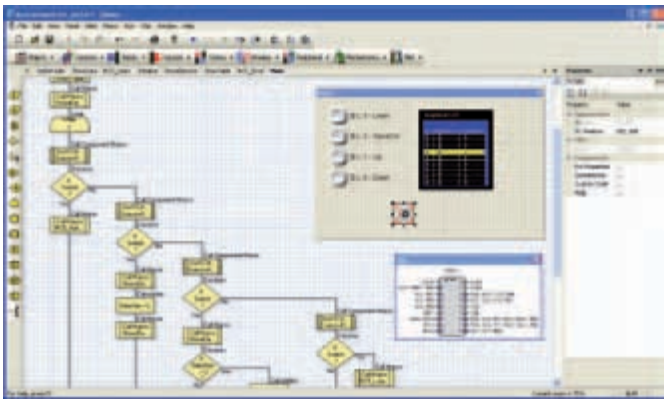


Figura 9. El programa Flowcode.

Programa Flowcode

El programa Flowcode (ver **Figura 9**) hace uso del componente RC5 para desarrollar un transmisor de control remoto que puede aprender y mostrar códigos de otros mandos. La pantalla gráfica LCD de E-blocks (EB058), muestra una tabla de diez mensajes RC5, los cuales tienen como valor por defecto 0 para las direcciones y valores de datos de 0 a 9 (TV principal, botones del 0 al 9). El programa está controlado por cuatro botones de un E-block de Conmutación (EB007), conectado al mismo puerto que la pantalla (Puerto BL). El mensaje destacado es transmitido por el E-block RC5 (EB060) cuando se presiona el botón Send (Interruptor 2). La barra de selección amarilla puede ser movida hacia arriba y abajo usando los botones 1 y 0, respectivamente. El modo de aprendizaje se activa presionando el botón 3. En ese momento aparece un menú de pantalla que muestra tres opciones. Los botones 0 y 1 pueden usarse para resaltar la opción requerida; el botón 2 se usa para confirmar la selección. Si seleccionamos el modo de aprendizaje se muestra la tabla principal, con la barra de selección en rojo. Los detalles del mensaje seleccionado serán sobrescritos por cualquier mensaje válido recibido por el E-block RC5, cambiando la barra de selección a su funcionamiento normal en color amarillo y restaurando el modo normal de trabajo. El modo de aprendizaje también puede ser cancelado presionando el botón 2. El enlace por hilo que se muestra en el zócalo de U1 "enruta" el terminal RD0/INT1 al conector PORTA/Miscellaneous donde es usado para la interrupción de recepción. Las señales RC13 y RC14 también están disponibles en este conector y son usadas para controlar la transmisión y la modulación. Puesto que este programa ha sido escrito para aprender y recuperar comandos del mando a distancia, deberíamos ser capaces de modificarlo para realizar cualquier función que necesitemos. El programa Flowcode puede ser descargado libremente de la página web Elektor [1].

Conclusión

La codificación y descodificación de las señales infrarrojas RC5 no pueden ser descritas como una 'tarea trivial', pero en este artículo hemos mostrado que el uso de una máquina de estado dentro de Flowcode puede simplificar bastante la tarea de escribir el programa.

(110315)

Internet Link:

[1] www.elektor.com/110315

Dentro del USB

En lo que a comunicación entre el PC y dispositivos externos se refiere, el Universal Serial Bus es el estándar indiscutible. Su mayor logro es la sencillez para el usuario final, en el caso más simple sólo hemos de conectar el cable. ¡Pero a los lectores de Elektor no les basta con esto! Aquí presentamos las bases principales de este protocolo.

Guy Weiler (Luxemburgo)

El protocolo USB para comunicar un PC (Host) y un dispositivo (Device) consta de varias capas. La unidad de transferencia más compleja (solicitud, preparación y ejecución de la comunicación) se conoce como USB-Transfer. Aparte de la única excepción (el llamado Control-Transfer) el resto de datos transferidos llevan un único sentido.

Una transferencia consiste en una o más transacciones, cuya característica es que han de ser ejecutadas a modo de tren. Cada transacción se compone a su vez de un paquete de token (Header), un paquete de datos opcional y un paquete de status (verificación del control). Los paquetes están divididos, según el tipo de paquete le corresponde una ID particular en la que se encuentra la dirección del receptor y el control de errores (ver la **figura 1**).
¡Y ahora viene lo bueno!

Tipos de transferencias

El Universal Serial Bus es, como su propio nombre indica, utilizable universalmente, sin embargo existen cuatro tipos de transferencia distintos. La Control-Transfer ocupa un lugar especial, y depende de las necesidades de cada dispositivo (ver abajo). Las otras tres están referidas a las distintas especificaciones de la comunicación:

- **Bulk-Transfer:** para transferir grandes cantidades de datos sin una velocidad garantizada (por ejemplo un disco duro).
- **Interrupt-Transfer:** con ancho de banda garantizado para un volumen de datos pequeño (por ejemplo un teclado).
- **Isochron-Transfer:** con ancho de banda garantizado, pero sin corrección de errores (por ejemplo el streaming de audio).

Estas tres transferencias siempre son unidireccionales independientemente de los datos transmitidos. Se componen de unidades más pequeñas e indivisibles, las transacciones. Hay transacciones de entrada (IN) que envían datos del dispositivo al PC y otras de salida (OUT), que mueven datos del PC al dispositivo (IN/OUT visto desde el lado del PC).
Las transacciones se dividen a su vez en tres paquetes. El paquete de token especifica el tipo de transacción (IN/OUT). El paquete de Data, como su nombre indica, contiene los datos, mientras que el de Handshake (protocolo) devuelve el acuse de recibo por parte del receptor. Aquí existen las siguientes posibilidades: “éxito” (ACK) o “estoy ocupado, por favor espera” (NAK).

Control-Transfer

Parece mentira que así sea, pero la naturaleza del bus USB hace que la comunicación entre PC y dispositivo sea casi completamente automática. El proceso mediante el cual el host asigna una dirección al dispositivo, se pregunta toda la información a este, se carga el driver correcto y con ello se selecciona la configuración se llama enumeración (“Enumeration”). Además, cada dispositivo no tiene por qué responder utilizando una sola dirección, sino que puede servirse de varios “puntos terminales” en los cuales entrega o recoge datos (ver el cuadro). Nuevamente, para saber el número y tamaño de dichos puntos ha de preguntar al host/PC.
La citada enumeración del dispositivo se lleva a cabo mediante un tipo especial de transferencia bidireccional, la llamada Control-Transfer (siempre en el punto terminal 0 del dispositivo). Mediante las Control-Transfers también puede controlarse el dispositivo. Al igual que los otros tipos de transferencias, dicha Control-Transfer se compone a su vez de transacciones, normalmente:

1. Una transacción de setup.
2. Opcionalmente una o más transacciones de datos.
3. Una transacción de status.

Cómo funciona este protocolo

Existen Control-Transfers de lectura y de escritura. Tanto si el host quiere escribir datos como leerlos del dispositivo, se le informa mediante una transacción de setup.

Las transacciones

La transacción de setup consta, como todas las transacciones, de paquetes de token, data y handshake. El PC envía los paquetes de token y data, el dispositivo responde con el paquete de handshake (**figura 2**).

Toda la información de la pregunta en el paquete de 8 bytes Setup-Data. El primer byte *bmRequestType* determina entre otros la dirección de la pregunta (Bit7=0 significa que va del PC al dispositivo, Bit7=1 al contrario) y si se trata de una pregunta estándar o específica del fabricante.
Los otros bytes constan de los campos *bRequest* (1), *wValue* (2), *wIndex* (2) y *wLength* (2).

Primero echaremos un vistazo a la transferencia descrita, observemos la figura 2. Tras recibir los datos dentro de la transacción de data el dispositivo ya puede enviar el acuse de recibo (ACK), la solicitud de espera al PC (NAK) o la orden de cancelar (STALL). Si los paquetes de token o data fallan, se ignoran. Como ya hemos dicho, se suceden muchas transacciones de datos consecutivas (en la gráfica sólo se muestra una).
El dispositivo confirma en la transacción de status si todo ha ido bien. Ahora se intercambian los roles, el dispositivo envía datos de (status) y el host responde con un paquete de handshake. Si la transferencia ha tenido éxito, el dispositivo envía un paquete con "ZeroLength" al PC. El PC responde con ACK. Si ocurre un error en el punto terminal, el dispositivo envía STALL. Si está ocupado todavía, envía NAK.
Al leer la Control-Transfer (**figura 3**) el dispositivo puede reaccionar dentro de la transacción de datos de tres formas al token IN del host. El dispositivo puede suministrar los datos; si está ocupado, entonces

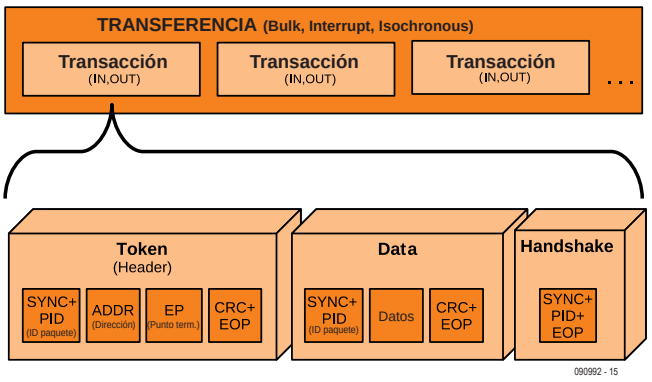


Figura 1. Las transferencias USB constan de transacciones indivisibles, que a su vez se componen de un paquete de token, uno de datos y otro de Handshake (aquí no se incluyen las Control-Transfers especiales).

envía NAK. Ocurre un error en el punto terminal, en tal caso envía STALL. Si ocurre un error en otra parte, entonces el token es ignorado y no se devuelve nada.
Mientras que en la transacción de status el PC devuelve un paquete de "ZeroLength" si se reciben correctamente los datos, por parte del dispositivo ha de confirmarse de nuevo mediante un paquete de handshake ACK.

Descriptores

Durante la enumeración se solicitan desde el PC mediante preguntas estándar (*standard requests*) muchos de los llamados descriptores. Estos descriptores del dispositivo están ordenados de forma jerárquica [1]. Cada dispositivo debe contar al menos con cuatro descriptores:

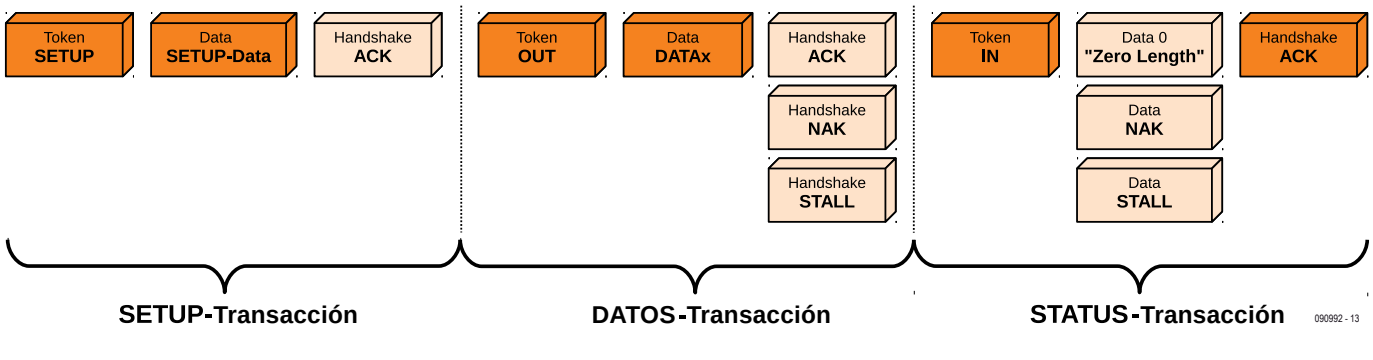


Figura 2. A cada Control-Transfer (aquí descritas) le precede una transacción de setup. Después opcionalmente le siguen más transacciones de datos y una de status.

El descriptor de dispositivo			
Nombre del campo	**Byte**	**Valor en la librería AVR [3]**	**Descripción**
bLength	1	18 (0x12)	Tamaño del descriptor en bytes
bDescriptorType	1	1	Descriptor de dispositivo = 1 (constante)
bcdUSB	2	0x0110	USB_Spec1_1
bDeviceClass	1	0xFF	Código de clase (específico del fabricante = 0xFF, ver [6])
bDeviceSubClass	1	0xFF	Subcódigo de clase (específico del fabricante = 0xFF)
bDeviceProtocoll	1	0xFF	Código de protocolo (específico del fabricante = 0xFF)
bMaxPacketSize	1	8	Tamaño máximo de paquete en el punto terminal 0 (EP0_FS)
idVendor	2	0x03eb	Código de Atmel en usb.org
idProduct	2	0x0001	ID de producto arbitraria
bcdDevice	2	0x0001	Número de lanzamiento del dispositivo
iManufacturer	1	1	Índice para el descriptor de string del fabricante
iProduct	1	2	Índice para el descriptor de string del producto
iSerialNumber	1	3	Índice para el descriptor de string del número de serie
bNumConfigurations	1	1	Número de configuraciones posibles

El descriptor de configuración			
Nombre del campo	**Byte**	**Valor en la librería AVR [3]**	**Descripción**
bLength	1	9	Tamaño del descriptor en bytes
bDescriptorType	1	2	Descriptor de configuración = 2 (constante)
wTotalLength	2	39 (0x27)	Longitud del descriptor de configuración y de los de todos los puertos y puntos terminales utilizados
bNumInterfaces	1	1	Número de puertos
bConfigurationValue	1	1	Número para seleccionar esta configuración (no debe ser cero, en caso contrario el dispositivo estará en estado no configurado)
iConfiguration	1	0	Índice para el descriptor de string de esta configuración (0 = sin texto)
bmAttributes	1	0x80	Bit7 = 1 alimentación por el bus, Bit6 = 1 alimentación propia, Bit5 = 1 Remote Wakeup
bMaxPower	1	50	Corriente máxima consumida por el bus en saltos de 2 mA

El descriptor de puerto			
Nombre del campo	**Byte**	**Valor en la librería AVR [3]**	**Descripción**
bLength	1	9	Tamaño del descriptor en bytes
bDescriptorType	1	4	Descriptor de puerto = 4 (constante)
bInterfaceNumber	1	0	Número de puertos
bAlternateSetting	1	0	Número de configuraciones alternativas a elegir
bNumEndpoints	1	3	Número de puntos terminales exceptuando el punto 0
bInterfaceClass	1	0xFF	Código de clase (específico del fabricante = 0xFF)
bInterfaceSubClass	1	0xFF	Subcódigo de clase (específico del fabricante = 0xFF)
bInterfaceProtocol	1	0xFF	Código de protocolo (específico del fabricante = 0xFF)
iInterface	1	0	Índice para el descriptor de string de este puerto (0 = sin texto)

El descriptor de punto terminal			
Nombre del campo	**Byte**	**Valor en la librería AVR [3]**	**Descripción**
bLength	1	7	Tamaño del descriptor en bytes
bDescriptorType	1	5	Descriptor de punto terminal = 5 (constante)
bEndpointAddress	1	0x81	Bit7 = 1 (IN), Bit0 a 3 número de punto terminal (para otro, 0)
bmAttributes	1	2	Tipo de transferencia = Bulk (Contr. = 0, Iso. = 1, Int. = 3)
wMaxPacketSize	2	8	Tamaño del FIFO del punto terminal en bytes
bInterval	1	0	Intervalo del polling = 0 (se ignora para Bulk y Control), 1 para Iso, de 1 a 255 para Interrupt

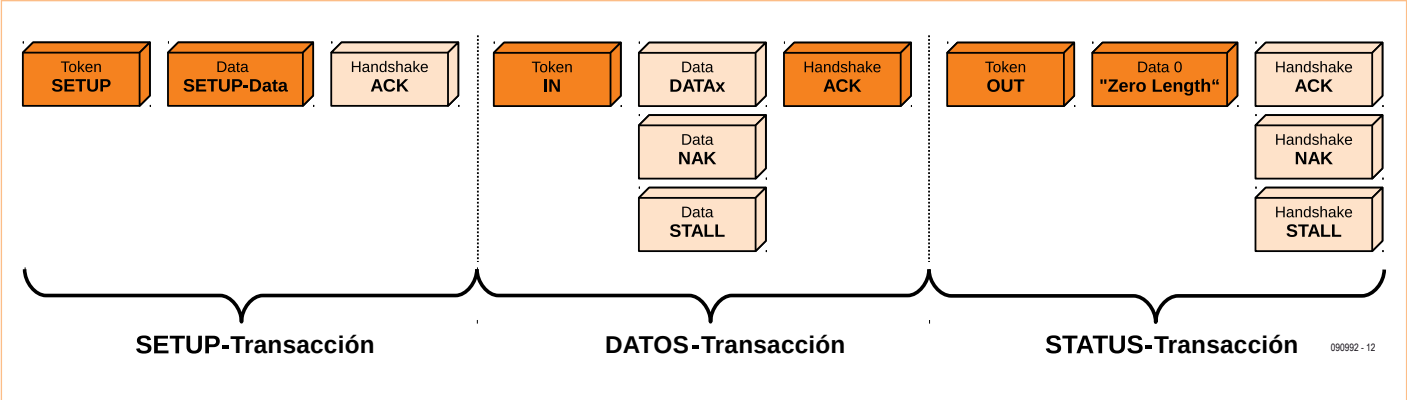


Figura 3. Lectura de una Control-Transfer.

- **Descriptor del dispositivo (device descriptor):** cada dispositivo cuenta con uno único.
- **Descriptor(es) de configuración (configuration descriptor):** un dispositivo puede contar con muchas configuraciones, por ejemplo una para ser alimentado a través del bus y otra para utilizar alimentación propia. Sólo una puede estar activa.
- **Descriptor(es) de puerto (interface descriptor):** el descriptor de puerto combina varios puntos terminales en un mismo grupo funcional. Pueden estar activos múltiples puertos a la vez (por ejemplo: puerto de fax, de impresora, y de escáner en el caso de impresoras multifunción).
- **Descriptor(es) de punto terminal (endpoint descriptor):** los descriptores de punto terminal describen dicho punto (excepto para el punto terminal 0). Es importante que tanto la dirección del punto terminal como el sentido, mediante el bit 7, estén debidamente especificados.

Los descriptores opcionales de string, no siempre son necesarios, pero suministran información que puede leerse adicionalmente. Estas cadenas están codificadas en Unicode (16 bits). Mediante Index 0 se fijan los idiomas compatibles.

La enumeración

Aquí sólo describimos los aspectos más importantes de una enumeración. Trabajaremos sólo con una configuración y un puerto.
Una vez conectado el dispositivo al bus, el PC determina mediante las tensiones de ambas líneas de señal si se trata de un sistema de Low o Full-Speed. El PC resetea el dispositivo (pone las dos líneas de datos a nivel bajo) y averigua si es un modelo High-Speed. Tras resetear nuevamente el dispositivo, estará listo para su acceso mediante el punto terminal 0 y la dirección 0.
Las preguntas (estándar) se implementan mediante Control-Transfers de lectura y escritura. Primero se preguntan los 8 bytes del descriptor de dispositivo de 64 bytes, para determinar el tamaño de paquete máximo del punto terminal 0 (éste puede ser de 8, 16, 32 y 64 bytes). Posteriormente el PC ejecuta un reset en el dispositivo y envía por fin la dirección de éste.
En la tercera transferencia se preguntan otros 18 bytes del descriptor de dispositivo; en la cuarta se obtienen 9 bytes del descriptor de configuración. Los datos revelan el tamaño total de los descriptores de configuración, puerto y punto terminal. La quinta transferencia pregunta a estos descriptores están todos activos.
Ahora el PC averigua mediante las IDs del fabricante y producto cual es el driver del dispositivo correspondiente.
La última transferencia en la enumeración inicializa y activa los puntos terminales de usuario (puntos 1-15).
Las preguntas por parte del PC que no sean compatibles con el firmware son respondidas por parte del dispositivo con STALL.

Más información

El autor de este artículo [2] ha desarrollado en colaboración con Jean-Claude Feltes (también redactor de Elektor) una librería básica USB para controladores AVR, que puede descargarse en [3] (la documentación está en alemán). En la parte del PC se utiliza la librería libusb [4].
Puede encontrar más información sobre el puerto USB en [5].

(090992)JN

Enlaces

[1] www.beyondlogic.org/usbnutshell/usb5.shtml#DeviceDescriptors
[2] weigu@weigu.lu
[3] www.weigu.lu/b/usb/key/vendor/index.html
[4] http://sourceforge.net/projects/libusb/
[5] www.elektor.es/090768
[6] http://es.wikipedia.org/wiki/Universal_Serial_Bus

Puntos terminales

Los datos se intercambian entre el PC y el punto terminal del dispositivo (*endpoint* EP). Un punto terminal es una memoria de datos (FIFO, buffer, banco de almacenamiento) en el propio dispositivo que generalmente consta de 8 a 256 bytes. Cada dispositivo tiene muchos puntos terminales, a los cuales se accede mediante la dirección de punto (0-15). Dicha dirección incluye en el bit 7 el sentido de comunicación de los datos (bit 7 = 0: punto terminal de OUT; bit 7 = 1: punto terminal de IN). El punto terminal de control 0 existe en todo dispositivo y es necesario por ejemplo para la enumeración. Como punto terminal independiente, es bidireccional. El USB 1.1 disponía de una memoria FIFO con un tamaño de 8 bytes. El número y tamaño posible de los otros puntos terminales es variable. Cada transacción está determinada por una dirección de dispositivo y punto terminal.

¡Que viene el bus! (6)

¡Tarjetas experimentales de Elektor y más!

Hemos alcanzado el siguiente escalón de nuestro bus gracias a las primeras tarjetas de Elektor con propósitos de diseño. Para disponer de un hardware lo suficientemente sólido se ha diseñado un nodo experimental con entradas tanto analógicas como digitales, así como un compacto conversor USB/RS485. Aparte, presentamos un sencillo sistema que garantiza una comunicación eficiente y segura a través del bus.

Jens Nickel

Motivados por nuestros primeros éxitos, seguimos adelante: en esta parte presentamos tarjetas de Elektor de fácil montaje, especialmente aptas para desarrollar aplicaciones del bus.

La base de nuestro “nodo experimental” es el circuito que ya presentamos en la anterior parte como “nodo de prueba”, y que sólo disponía de un LED y un botón de test. Sin embargo, esto era suficiente para comprobar que los mensajes se transmitían correctamente a través del bus. Tampoco es que fuese muy emocionante, aparte de que no permitía diseñar una aplicación real completa.

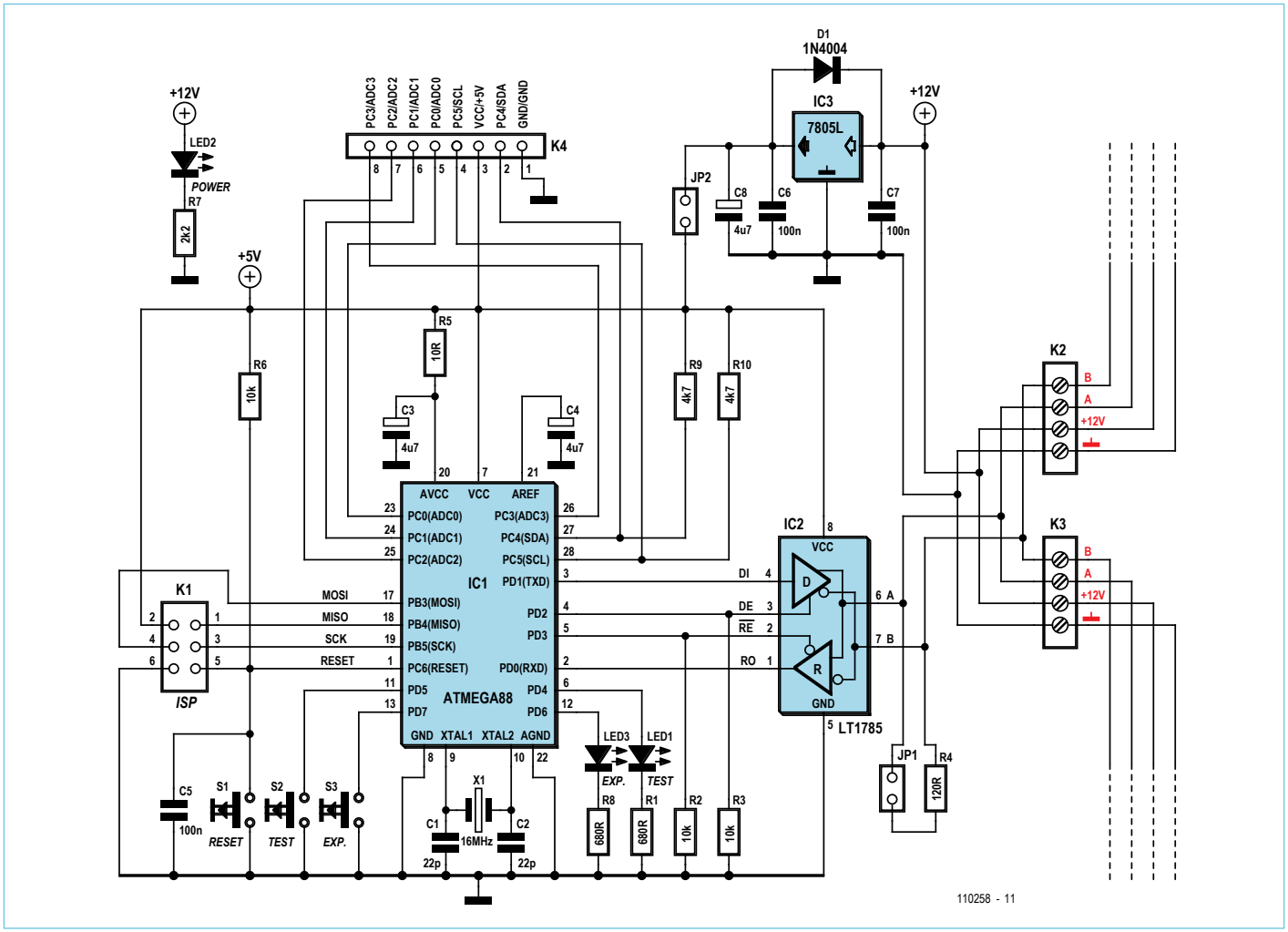
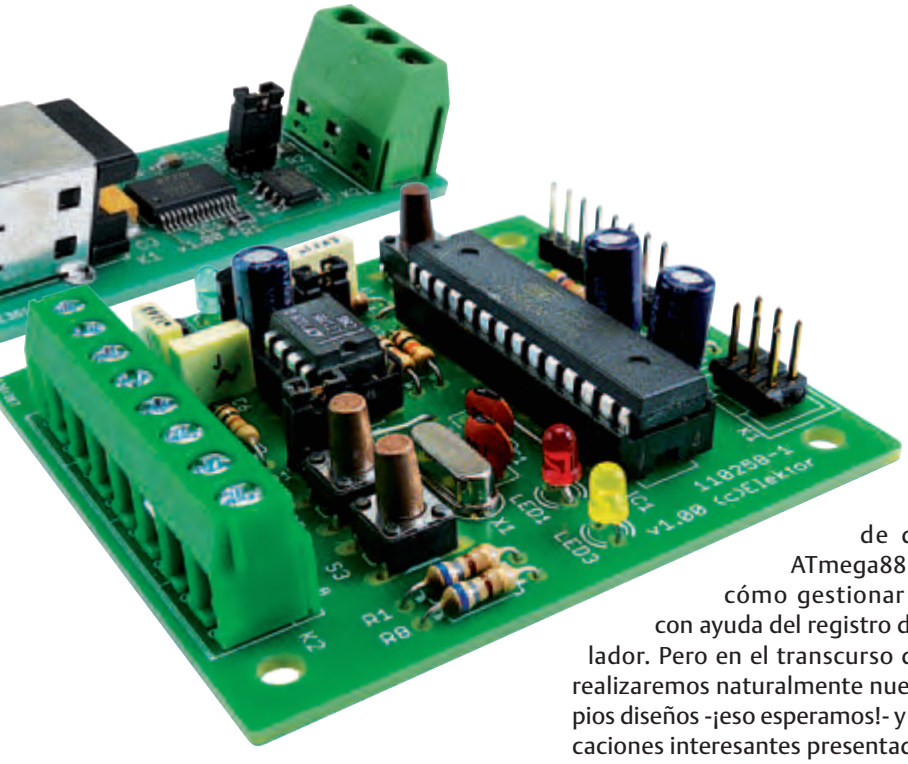


Figura 1. Esquema del nodo experimental.



Nodos experimentales

El esquema del nuevo participante del bus basado en el ATmega puede verse en la **figura 1**. Ya hemos explicado la funcionalidad del driver RS485 en entregas previas de la serie. Gracias a las clemas dobles de las que dispone, los nodos pueden cerrarse y al mismo tiempo hacer bucles en determinadas partes de la línea de bus. Como pequeño extra se ha añadido una resistencia de 120 Ω puenteable, que puede servir como terminación para las líneas RS485 del bus.

En los pines de puerto PD6 y PD7 hemos conectado un LED y un pulsador adicionales, que se nos antoja llamar "LED experimental" y "botón experimental". Sin embargo, la principal innovación la supone el conector K4, que dispone de seis pines de controlador adicionales, la alimentación del controlador y la masa del bus. Los pines de puerto PC0 a PC5 fueron cuidadosamente elegidos, ya que –como ocurre en muchos otros controladores– se encargan de multitud de tareas distintas. De este modo, pueden ser utilizados tanto como entradas o como salidas digitales. Mediante PC0 a PC3 también podemos medir cuatro tensiones analógicas (de 0 a 5 V) gracias a los conversores analógico-digital multicanal del controlador. Y finalmente, los pines PC4 (=SDA) y PC5 (=SCL) ofrecen conexión directa con el puerto I2C del ATmega. Según las especificaciones, las líneas SDA y SCL están provistas de resistencias de pull-up.

Ahora pueden conectarse sensores simples en K4, a modo de slaves I2C (como por ejemplo sensores de temperatura), así como otros componentes electrónicos. La hoja de datos del ATmega88 [1] revela cómo gestionar los pines con ayuda del registro del controlador. Pero en el transcurso de la serie realizaremos naturalmente nuestros propios diseños -¡eso esperamos!- y otras aplicaciones interesantes presentadas por los lectores.

Para propósitos de desarrollo los nodos experimentales también pueden alimentarse con K4 (en lugar de mediante el bus). En tal caso hemos de retirar el jumper JP2. En funcionamiento es más que recomendable conectar la masa del bus a su línea correspondiente y llevar esta a tierra (véase la última parte de la serie [2]).

Para redondear el conjunto, huelga decir que también se han cableado las entradas del controlador AVCC y AREF para medidas del ADC según cita la hoja de datos.

La tarjeta, de una sola cara, (**figura 2**) resulta fácil de montar; hemos de comenzar por los puentes de cable.

Conversor USB/RS485

Aquí podemos quedarnos con lo que ya teníamos: la tarjeta del conversor USB/RS485 se corresponde con el esquema del circuito del conversor ya mostrado en la anterior entrega (véase la **figura 3**). Ya que el chip de FTDI sólo está disponible en SMD, hemos optado por diseñar y equipar la tarjeta entera con componentes SMD (**figura 4**). En mi opinión se trata del conversor USB/RS485 más compacto y simple presentado en Elektor por el momento. Es ideal para nuestros propios diseños de bus, por ejemplo basándonos en lo que ya contamos y el software de PC presentado previamente. Hemos de advertir de una pequeña limitación, y es que el desarrollo del ElektorBus todavía no está cerrado. Por el momento ni siquiera sabemos si necesi-

Lista de materiales del nodo experimental

Resistencias:
R1, R8 = 680 Ω
R2, R3, R6 = 10 kΩ
R4 = 120 Ω
R7 = 2k2
R9, R10 = 4k7

Condensadores:
C1, C2 = 22 pF
C3, C4, C8 = 4µ7
C5, C6, C7 = 100 nF

Semiconductores:
D1 = 1N4004
IC1 = ATmega88-20PU
IC2 = LT1785
IC3 = 78L05

Varios:
X1 = oscilador de cuarzo a 16 MHz
S1, S2, S3 = pulsador
JP1, JP2= Jumper
LED1 = LED 3 mm, rojo
LED2 = LED 3 mm, verde
LED3 = LED 3 mm, amarillo
K1 = conector tipo pin header de 2x3 (RM 2,54 mm)
K4 = conector tipo pin header de 1x82x3 (RM 2,54 mm)
K2, K3 = clema para montaje en tarjeta de 4 pines (RM 5,08 mm)
Tarjeta 110258-1 [3]

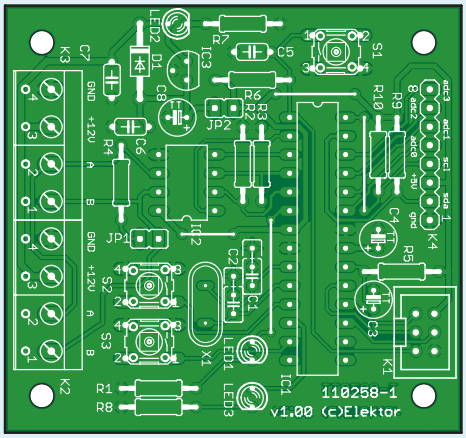


Figura 2. La tarjeta puede montarse completamente con componentes convencionales.

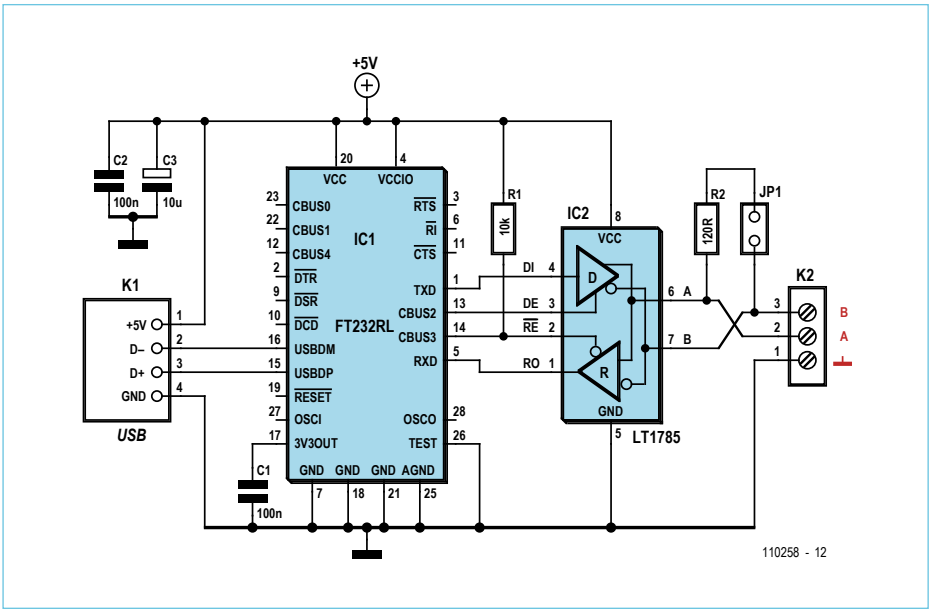


Figura 3. Esquema del conversor USB/RS485.

taremos una solución más inteligente entre el RS485 y el PC (ver más abajo). Actualmente con el conversor vamos bien servidos, y naturalmente podemos utilizarlo para otras aplicaciones RS485.

Ya que por ahora disponemos de tarjetas para varios experimentos, también nos gustaría contar con algo de software. El software de prueba presentado en la parte 5 [2] no incorporaba ningún mecanismo para evitar colisiones o garantizar una estabilidad mínima. En adelante presentamos un sencillo sistema que ofrece una comunicación por el bus eficiente y segura. No obstante, en nuestro grupo de mails acerca del bus aparecieron otras sugerencias, puede ver un pequeño resumen en el cuadro de texto.

Polling con el método Round-Robin

En los capítulos previos de la serie ya hemos explicado las ventajas e inconvenientes de llevar a cabo un muestreo cíclico (polling) en todos los nodos. Cada nodo es preguntado (uno por uno) con un "Elektor-Message" de 16 bytes exactos desde un planificador, y éste responde correspondientemente con un sólo mensaje. Un sistema así de simple tiene dos claros inconvenientes.

1. Si un emisor quiere comunicar algo, hemos de esperar a que le toque el turno en la serie. Por ejemplo, consideremos un interruptor en un sistema domótico, activado por un usuario. En el peor de los casos, el correspondiente mensaje se enviaría tras responder el resto de dispositivos del bus. A 9600 baudios y con 32 participantes en el bus esto podría llevar más de un segundo, podría resultar confuso para nuestros visitantes hasta que llegue a encenderse la luz.
2. Desarrolla demasiado intercambio de datos innecesario en el bus. Por ejemplo, no tiene sentido preguntarle a un interruptor cada segundo, si éste cambia de estado como mucho 10 veces al día.

El otro extremo

Para los nodos que envían datos muy de vez en cuando (como por ejemplo los interruptores), resulta más útil otro método: el interruptor debe enviar al ser pulsado sin tener en cuenta lo que hagan el resto de participantes del bus. Naturalmente, podrían darse colisiones, provocando una secuencia aleatoria de bits a través del bus. En lugar de un mensaje correcto de 16 bytes tendríamos más bytes con valores totalmente erróneos, y al final ambos mensajes "se pierden por el camino". Por ello, en los mensajes con sentido el receptor ha de responder con un acknowledge-message (acuse de recibo). Cuando esto falla, el emisor repite el procedimiento. Este sencillo método también tiene en cuenta si los mensajes se pierden debido a otras perturbaciones.

No obstante, este método también tiene sus inconvenientes. Si tenemos varios participantes en juego, enviando mensajes a la vez a través del bus, aparecerán muchas colisiones. Aparte, deberíamos disponer de un nodo de análisis, que reciba por ejemplo valores de la temperatura desde un nodo sensor, respondiendo cada vez con un mensaje de acknowledge, lo cual incrementa la carga en el bus y aumenta el peligro de darse colisiones.

Una buena mezcla

Cada uno por separado, tanto el lector Jürgen Lange como yo tuvimos la misma idea: ¡Combinemos lo mejor de ambos mundos! Sencillamente cada cierto tiempo cambiamos entre el polling (el nodo sólo puede hablar si es preguntado) y el otro extremo (el nodo puede enviar datos sin preocuparse del resto).

Una vez los nodos han sido preguntados uno tras otro, teniendo que transmitir algo regularmente (como ocurre con los

Lista de materiales del conversor USB/RS485

Resistencias:
R1 = 10 kΩ (0805)
R2 = 120 Ω (0805)

Condensadores:
C1, C2 = 100 nF
C3 = 10 µF/16 V (6032)

Semiconductores:
IC1 = FT232RL
IC2 = LT1785 (SO-8)

Varios:
JP1 = jumper
K1 = conector USB tipo A
K2 = clema para montaje en tarjeta de 3 pines (RM 5,08 mm)

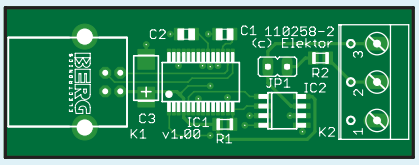


Figura 4. El conversor USB/RS485 de Elektor se suministra montado y probado.

Tarjeta 110258-2 [3]
o
Tarjeta ya montada y probada 110258-91 [3]

Alternativa: soluciones a nivel de bit

Diseñando conceptualmente nuestro nodo experimental, he de decir que es la primera vez en mi vida que me gano el sueldo como desarrollador de circuitos ;-). Pero básicamente he hecho de “niñera”, actuando de moderador y supervisando nuestra lista de mails. Como ya se dijo en la anterior entrega de la serie: hay un grupo de experimentados desarrolladores de buses que están luchando de forma vehemente contra los errores de colisión a nivel hardware. El experto en CAN John Dammeyer puso un método en juego, en el cual cada emisor esperaba un tiempo “Space-Time” de 12 bits, antes de poder enviar ningún mensaje. Ya que este escucha mediante el bus, puede reconocer si otros participantes están hablando a través de él. En cuanto a otras propuestas para la detección de las colisiones, como discutimos en grupo, hemos de servirnos del llamado Bit-Banging (así como la manipulación de pines de puertos UART mediante el software).

En cuanto a la re-sincronización de los participantes del bus mediante un mensaje de inicio, existen distintos métodos para ello. El lector de Elektor Walter Trojan se decantó por el llamado modo en 9 bits, el cual permiten el ATmega88 y otros controladores más (conocido como Atmel Multi-processor Communication Mode, ver [1]). En éste se envían, en lugar de 8 bits, 9 en un mismo frame, y el receptor puede reconocer por ejemplo cuándo se trata de un byte de dirección o de datos.

Lamentablemente, aquí no podemos describir al detalle todas las ventajas e inconvenientes de los métodos citados. Sólo queda añadir que estos inteligentes métodos tienen una desventaja principal para mí: son restrictivos en cuanto a la cifra de procesadores y plataformas permitidas, limitando así el número de participantes del bus. No todos los controladores disponen de un modo de 9 bits, y el bit-banging por software de PC se me antoja –aunque no quede completamente descartado– una forma no demasiado elegante. Pero, ¿y si ahora queremos transmitir nuestros mensajes a otras redes (por ejemplo una inalámbrica)? Los bytes son unidades de información, mediante las cuales puede controlarse un sistema de forma práctica con words de 9 bits y con bits individuales, aunque resulta más complicado.

Como puede imaginar, actualmente la discusión sólo puede tomar un camino posible... ante todo quería contar con la posibilidad de controlar nuestro bus RS485 directamente desde el software de PC, pero me acabaron regañando por ello. Un PC con Windows, debido a las variaciones en el timing de dicho sistema operativo de Microsoft, no es la mejor opción; aparte, hemos de encontrar la manera de utilizar un conversor USB/RS485 así como una puerta de enlace USB/RS485, sirviéndonos de un controlador adicional capaz de gestionar la comunicación en el bus.

En resumidas cuentas: me gustaba la posibilidad de que nuestros mensajes de 16 bytes pudieran servir dentro de los límites de otras plataformas y redes, pero resultaba demasiado bonito y tuve que darme por vencido al respecto. Para que siguiera siendo flexible en la medida de lo posible, no se trataba de mantener una comunicación segura y robusta en cuanto a hardware y a nivel de bit, sino a nivel de la seguridad en los mensajes (ver texto). Lógicamente, ¡tus opiniones son más que bienvenidas!

sensores de temperatura), el planificador libera el bus para poder enviar sin necesidad de ser preguntados previamente. Así es como envían datos los nodos que pocas veces tienen algo que decir (por ejemplo los interruptores). Esta fase libre en el bus (*FreeBusPhase*) ha de tener un tiempo limitado, de modo que no se envíe nada por error o una vez que el polling ha vuelto a empezar.

Junto a lo que yo llamo poner en práctica el *Hybrid-Mode* también se ha ampliado el software de la anterior edición [2]. Primero, se han implementado todas las funciones descritas, pudiendo integrarlas luego posiblemente en una librería; más abajo puede ver una pequeña aplicación. Puedes encontrar todo, como siempre, en la página web de Elektor, código fuente inclusive [3].

Planificador

En este caso el PC hace las funciones de planificador. Se direcciona como emisor 0, de modo que las instrucciones sean reconocibles desde cualquier nodo. El planificador almacena en un array (de 0 a *intPolledNodesMax*-1) las direcciones de los nodos, las cuales son preguntadas siguiendo un ciclo constante. Naturalmente también es posible “hacer polling” con mayor frecuencia en nodos que sean especialmente comunicativos.

Para dicho polling en un nodo el planificador envía una pregunta (*SchedulerRequest*), siendo la dirección del destinatario la del nodo en cuestión. Después el planificador espera al mensaje enviado desde la dirección correspondiente (*ResponseMessage*), que puede provenir de un nodo arbitrario. Ahora le toca al siguiente nodo de la lista y así en adelante. Si uno de los nodos está ausente, el lógicamente el sistema completo entrará en estado de espera. Por ello, al comenzar con las *SchedulerRequests* ha de iniciarse un timer, que vuelva a empezar cada vez que pasamos al siguiente nodo a preguntar. La variable *intFreeBusCycle* tiene una considerable importancia. Determina

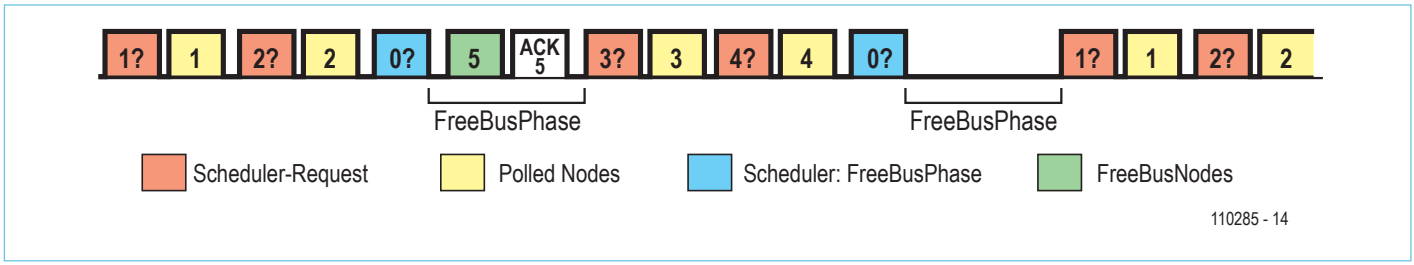


Figura 5. En el modo híbrido se intercambian las fases de Polling con otras en las que el bus está libre. Durante la FreeBusPhase pueden originarse colisiones, con lo que los receptores tendrán que responder con acuses de recibo (ACK).

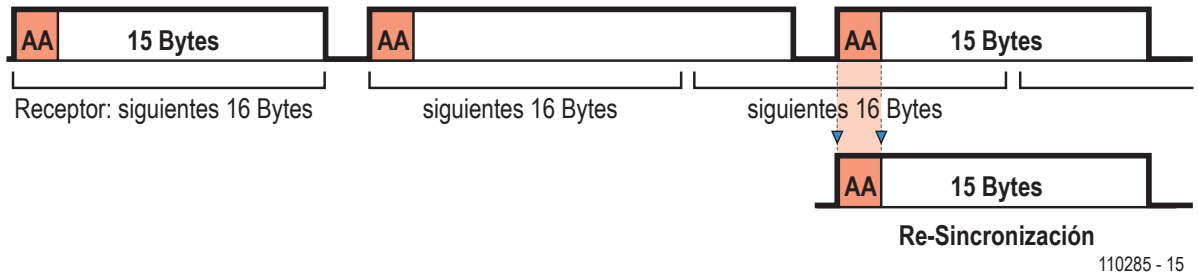


Figura 6. En las colisiones se forman grupos de bytes que pueden superar los 16 bytes. Después de esto los participantes del bus han de sincronizarse de nuevo al iniciar los mensajes otra vez, y tras buscar el valor AAhex en el flujo de bytes.

tras cada cuántos nodos preguntados toca intercalar una *FreeBusPhase*. Por ejemplo, un 2 significa que tras hacer polling en dos nodos (según el orden del array) será posible enviar mensajes libremente (véase la **figura 5**).
Para introducir una *FreeBusPhase* el planificador envía un *FreeBusMessage* especial, simplemente a la dirección de receptor 0 (los bytes de datos no se utilizan). Después el planificador espera el tiempo especificado por *intFreeBusTime* (dado en milisegundos) hasta que se reestablece la rutina de polling.

Firmware

Por el lado del nodo del controlador, se ha utilizado un software en BASCOM basado en el de la última entrega. Por recomendación de Günter Gerold en nuestra lista de mails sobre el bus he cambiado por completo la rutina de interrupciones, de modo que ahora se sale de ella tras recibir un carácter. Al igual que antes, los caracteres recibidos se almacenan en un array de un byte. Tras recibir 16 caracteres, la rutina averigua si la dirección del receptor es distinta de 0 (*FreeBusMessage*). Posteriormente se activa el *ReceivedEventFlag*. Esto indica que se ha recibido un mensaje, y por lo tanto debe ser procesado en el bucle principal.
Hemos de tener en cuenta que un nodo (con dirección propia) al cual el planificador ha preguntado directamente ha de responder en el momento. A un *FreeBusMessage* han de responder sólo los nodos a los que se ha preguntado (*FreeBusNodes*), y en caso de que desde el controlador se haya lanzado un mensaje para solicitar el estado (*SendEventFlag*=True). Si se envía un mensaje de este tipo tras la entrada de un *FreeBusMessage*, podemos permitirnos evitar los timers que supervisaban el *FreeBusTime* en el controlador.
En la EEPROM del controlador se guarda de qué tipo de nodo se trata (*PollingStatus*). De este modo se utiliza el mismo firmware en ambos casos.
Al enviar los mensajes del controlador utilizamos el bit aún libre del byte de modo (*ElektorMessageProtocol*, ver la **figura 7**). Bit 0=0 significa que se trata de un *ResponseMessage*, y que en principio no pueden darse colisiones de ningún tipo. Bit 0=1 significa que el mensaje se transmite en modo no seguro *FreeBusPhase*; en tal caso el receptor ha de responder con un acknowledge-message. Si falta éste significa que ha habido una colisión. El emisor tendrá que enviar nuevamente el mensaje y esperar por lo menos hasta el siguiente *FreeBusPhase*. Para que no se dé otra colisión, ha de asegurarse que el segundo emisor envíe siempre de nuevo en esta fase. Esto puede lograrse mediante un sistema muy simple: cada emisor cuenta con un número x distinto de *FreeBusPhases*. Esta cifra x (*FreeBusPriority*) se almacena de forma estática en la EEPROM al igual que la dirección propia del nodo. Durante los desarrollos posteriores el software tendrá que ser ampliado, de modo que tanto la dirección, como el *PollingStatus* así como la *FreeBusPriority* puedan ajustarse de forma dinámica.
En el transcurso de las posibles colisiones pueden aparecer grupos de bytes en el bus más largos de 16 bytes. Por ello, hace falta un mecanismo que se sincronice de nuevo al comienzo del siguiente mensaje. En esta versión del software sólo hemos solucionado parcialmente el problema de la re-sincronización: se busca en el propio flujo de datos entre los bytes el valor AA_{hex}, el cual se toma como inicio de un nuevo mensaje (**figura 6**). Por el momento tenemos que conformarnos con la limitación de confiar en que dicho byte no aparezca en el paquete de datos transmitidos (el control de errores por CRC todavía no es siquiera posible).

Pequeña aplicación

Ahora urge una pequeña aplicación con la que a la vez también podamos probar nuestro primer nodo experimental. Para simular valores de medida muestreados continuamente, he conectado un potenciómetro de 100 kΩ en K4 (ADC0/5V/GND). He combinado los dos nodos de prueba que Günter diseñó y montó para la anterior entrega.

El nodo experimental recibió la dirección "02" y un *PollingStatus* "01", grabado en la EEPROM. A los dos nodos de prueba les adjudiqué las direcciones "01" y "03", un *PollingStatus* de "00" y una *FreeBusPriority* de "01" o "02".

Por simplicidad he mantenido el mismo firmware para los tres nodos. Al pulsar en el botón de test se activa el *SendEventFlag*, el LED de test cambia de estado y se actualiza el estado de dicho LED en un *LEDbyte*, que se transmite en el próximo envío justo después del primer byte de datos de nuestro mensaje (figura 7). El *PollingStatus* determina si hace falta preguntar adicionalmente al ADC0 (aparte, no debemos olvidarnos de informar al ATmega88 que debe utilizar AREF como tensión de referencia, véase el código fuente). Los 10 bits del resultado del ADC se dividen en dos bytes, que se transmiten posteriormente como segundo y tercero byte de datos. Para que no haya ningún AA_{hex} en los datos, el byte *ADClow* tiene en lugar de los 8 bits inferiores sólo 7 de ellos, mientras que el byte *ADChigh* tiene los 3 bits superiores (más significativos). El PC sirve bien como receptor de los mensajes, mostrando el estado de los tres LEDs y el valor de los ADCs (debidamente convertido). Para que la dirección del planificador (=0) no se confunda con la del PC, a éste se le ha asignado una segunda dirección (=10). Desde esta segunda dirección también pueden enviarse mensajes de acknowledge a los nodos 1 y 3 (dirección del receptor = dirección del nodo, dirección del emisor =10, primer byte de datos=16+*LEDbyte*).

Todo depende del timing

Tras algunos intentos, se fija un *FreeBusTime* de 50 a 70 ms como suficiente, aunque yo he estado trabajando de un modo un poco "sucio". Ya que mi pequeña aplicación en el PC es al mismo tiempo planificador y receptor de los mensajes del controlador, se le permite enviar un mensaje de acknowledge justo después de la *FreeBusPhase* (de forma asíncrona), y a los nodos sólo se les vuelve a preguntar después de esto. Sin embargo, como regla general las rutinas del planificador y el envío de acuses de recibo han de desarrollarse síncronamente. Entonces debemos cuadrar el envío de mensajes y los acuses de recibo en una *FreeBusPhase*, como se muestra en la figura 5.

Hasta que el software funcionó sin errores, todavía fueron necesarias muchas más medidas. He aquí uno de los bugs que más dolor de cabeza me dio hasta que fui capaz de dar con el error: en el firmware del controlador, los mensajes enviados se perdían continuamente. El motivo era el siguiente: durante la rutina de interrupciones no se buscaba correctamente la dirección del receptor. Si durante la ejecución del bucle principal (botón, ADC) se envían no más de dos mensajes a través del bus, el segundo mensaje sobrescribe las variables utilizadas para analizar el primero. Hemos de buscar en la rutina de interrupciones al menos la dirección correcta del receptor y "aceptar" únicamente los mensajes correspondientes al siguiente procesamiento. Entonces dispondremos de más tiempo para las verdaderas aplicaciones de control (como por ejemplo el procesado de datos de medida). El modo híbrido tiene la ventaja de que no se pueden enviar dos mensajes uno tras otro al mismo receptor.

Conclusión

El hecho de que en el PC dispongamos de planificador, participantes del bus y representación de los resultados hace que el software sea bastante redondo. En un futuro sería más fácil realizar un planificador en un microcontrolador. Una posibilidad sería utilizar un controlador y supervisarlo mediante el adaptador recién diseñado USB/RS485 (requerido por varios participantes de la mail-list del bus, ver el cuadro).

El mecanismo de re-sincronización debe ser extendido hasta aquí, chequeando si se encuentran bytes con el valor AA_{hex} o si se cumple el CRC, lo cual ya ha tenido varias sugerencias.

Todavía hemos de lograr que los nodos puedan presentarse por su cuenta al planificador (y ser direccionados dinámicamente). Y

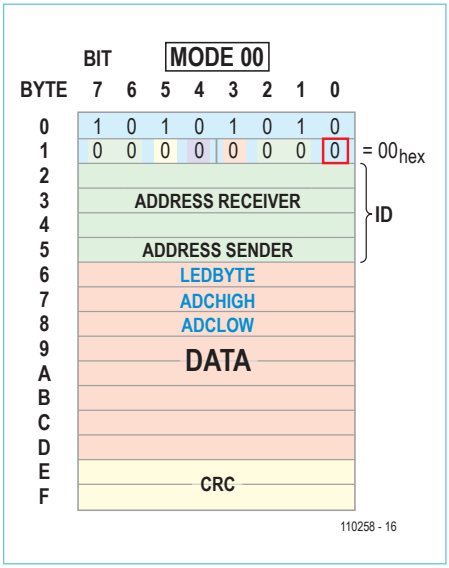


Figura 7. Nuestra pequeña aplicación se sirve de 3 bytes de datos del ElektorMessageProtocol (CRC todavía sin implementar). El bit 0 del byte de modo muestra si el mensaje proviene de el nodo preguntado (=0, en caso contrario =1).

por último, y no por ello menos importante, en el transcurso de las próximas entregas se desarrollará una aplicación con un uso real –en la cual, queridos lectores, ¡sois bienvenidos a tomar parte en el desarrollo!

(110258)

¡Forma parte del desarrollo! ¡Cualquier consejo o sugerencia es bienvenido en el mail de nuestra redacción, redaktion@elektor.de!

Enlaces

[1] www.atmel.com/dyn/resources/prod_documents/doc2545.pdf

[2] www.elektor.es/110225

[3] www.elektor.es/110258

Computerscoop

La nueva serie PicoScope 3000 en la práctica

Harry Baggen (Elektor Holanda)

Pico Technology ha renovado por completo su serie de osciloscopios USB de categoría media. La nueva serie PicoScope 3000 ofrece muchas cosas, como una frecuencia muy elevada de muestreo de osciloscopio alimentado a través de USB de 500 Msamples/s y un generador de formas de onda arbitrarias/generador de funciones. ¿Cuánto gusta este osciloscopio en la práctica? Nos pusimos a trabajar con un 3206B.

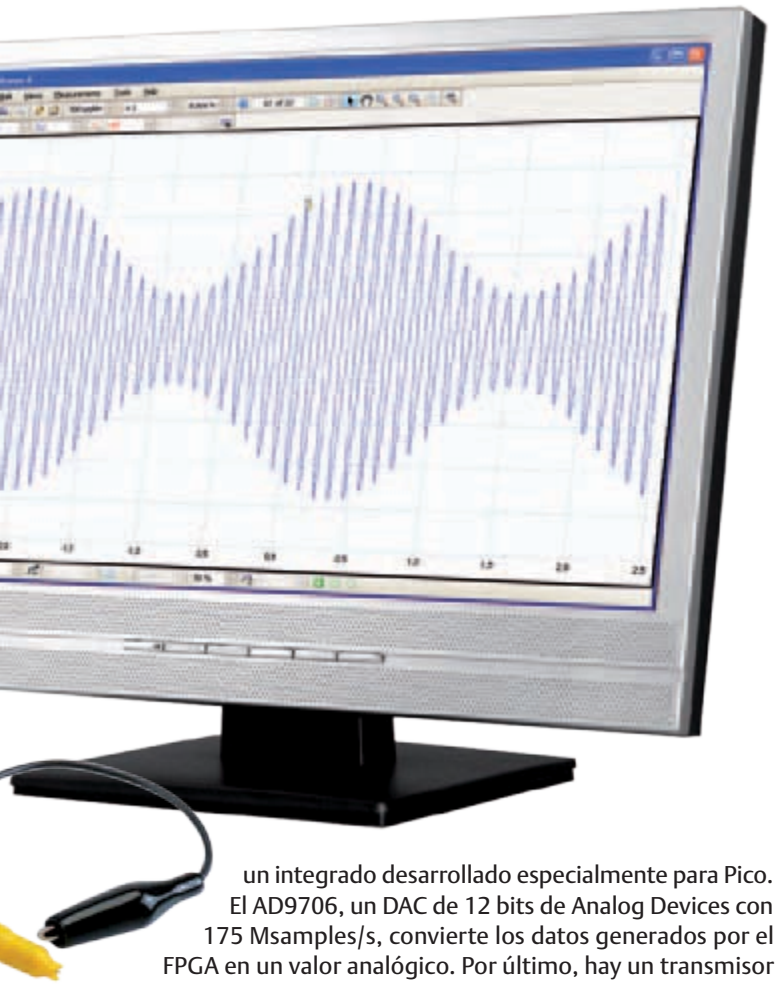
Los osciloscopios USB son cada vez más potentes y polifacéticos. A un electrónico le resulta cada vez más difícil resistir la tentación de adquirir un osciloscopio USB en vez de una versión autónoma. No tiene porque ser para ahorrar dinero, sino por las ventajas que ofrece una versión USB. Es un aparato de pequeñas dimensiones y tiene una pantalla (de ordenador) magnífica, algo con lo que un osciloscopio normal no puede competir. También se puede utilizar un osciloscopio USB en combinación con un portátil durante bastante tiempo alimentados por baterías, y además dicha combinación se aísla galvánicamente de la red eléctrica de forma automática. Y muchas veces el software incluido ofrece un montón de posibilidades adicionales, que no encuentras en una versión autónoma comparable. Pero un osciloscopio tiene también sus puntos fuertes. Este instrumento puede ofrecer ventajas especialmente a frecuencias de muestreo más altas, porque todo el hardware puede ser ajustado a la velocidad de procesamiento necesaria. En un modelo USB estás limitado a la velocidad de transmisión de datos entre el osciloscopio y el PC. Esto se puede evitar, por ejemplo, dotando al hardware de una gran cantidad de memoria ultrarrápida. Pero de esta manera el hardware es cada vez más extenso, lo que hace que la diferencia de precio sea más pequeña. Pero, como hemos dicho anteriormente, ¡un osciloscopio USB tiene varias victorias ganadas importantes en la mano!

La nueva serie 3000 de Pico

Pico Technology es un fabricante inglés especializado en el diseño y producción de aparatos USB de medición. ¡Tiene ya una experiencia de 20 años en esto! La oferta de osciloscopios USB es muy amplia, desde los modelos más económicos de 150 euros hasta los ultrarrápidos de más de 15.000 euros. Hace poco ha renovado por completo su popular serie 3000 de la categoría media. Esta consta de seis modelos de 2 canales con precios que oscilan entre 500 y 1100 euros. Toda la serie tiene una velocidad de muestreo máxima de 500 Msamples/s, estos son, según Pico, los osciloscopios alimentados por USB más potentes que se pueden adquirir actualmente. El ancho de banda de entrada aumenta por pasos y precio desde 60 hasta 100 y 200 MHz. Además hay versiones A y B que llevan incorporadas otro generador de funciones. Los modelos A llevan un generador de funciones que proporciona varias formas de señales fijas, mientras que las versiones B incorporan un generador de formas de onda arbitrarias (AWG=Arbitrary Waveform Generator) con el que el propio usuario puede crear formas de señales. Todos disponen de un buffer de memoria con un tamaño de entre 4 hasta 128 Msamples, dependiendo del modelo. Pico nos envió el modelo más caro de la serie, un 3206B, con el que hemos estado midiendo durante varios días. He aquí nuestro informe.

Hardware

Para empezar no pudimos resistirnos a abrir la caja del PicoScope e inspeccionar su hardware. En la ***figura 1*** puedes observar la caja abierta, donde llama directamente la atención las secciones de entrada perfectamente protegidas. El corazón del osciloscopio consta de un potente FPGA Spartan-6 de Xilinx que se ocupa tanto del procesamiento de las señales de medición como de la generación de las señales del generador. Está conectado a un integrado de memoria DDR3 veloz para el almacenamiento de los muestreos de datos. De la digitalización de los dos canales de entrada se encarga un convertidor A/D cuyo número de serie no es visible. Se trata de



un integrado desarrollado especialmente para Pico. El AD9706, un DAC de 12 bits de Analog Devices con 175 Msamples/s, convierte los datos generados por el FPGA en un valor analógico. Por último, hay un transmisor USB de Cypress (CY7C68013A) que se encarga de la comunicación con el PC.

Software

Pico suministra el mismo software con todos sus osciloscopios USB, el cual no tiene ninguna limitación en lo que concierne a sus características. La única limitación la forma las posibilidades y especificaciones del hardware conectado. La aplicación por defecto es la pantalla del osciloscopio, en la que se muestran las señales en un color determinado sobre un fondo blanco en tiempo real. Hay un modo especial persistente, donde se emula una pantalla de osciloscopio destellante (con un fondo negro). Los ajustes de disparo son especialmente amplios. Con la ayuda de algunos botones puedes recorrer el buffer de la memoria en bloques y mostrar las señales interesantes. También dispone de un modo XY. Luego hay un analizador FFT que muestra continuamente un análisis de la frecuencia de la señal de entrada. Ahí puedes seleccionar entre varios tipos de ventanas de análisis (Hamming, Blackman, etc.). El software ofrece además la posibilidad de mostrar y analizar varios tipos de datos en serie, como son el I²C, RS232/UART, el SPI y el bus CAN. En este caso, se muestran al mismo tiempo tanto la señal original medida como los datos decodificados en pantalla. A través de un botón generador puedes hacer que aparezca el menú del generador incorporado, para después configurar la forma de onda y la tensión de salida deseadas. En la versión B de la serie 3000 dispones además de la posibilidad de crear tus propias formas de onda con la ayuda de una ventana gráfica (dibujar mediante el ratón la forma de onda) o un fichero csv.
Además hay varios extras y opciones, demasiados para poder comentar todos ampliamente, como es la posibilidad de crear un perfil especial de sonda, la creación de máscaras para mediciones, entre otras cosas, de tolerancias en procesos de producción, la realización de todo tipo de mediciones sobre las señales mostradas y la realización de funciones matemáticas.

La práctica

El PicoScope 3206B se suministra con dos sondas en un bolsa de protección, un cable USB, un CD de instalación y una guía de inicio rápido. La discreta caja de plástico tiene 4 buses BNC en la parte frontal y un conector USB en la parte posterior. La instalación del software desde el CD se hace en cinco minutos, esto transcurre de modo impecable. Después puedes empezar tras conectar el osciloscopio con el PC mediante el cable que se incluye.
En primera instancia, el programa causa una impresión sobria después de iniciarse, todos los elementos de control se encuentran en algunas barras en la parte superior e inferior de la pantalla del osciloscopio. ¡Llama la atención el gran formato de la pantalla, en comparación con los osciloscopios estándar del laboratorio! La verdad es que deberías empezar a leer el menú de ayuda en el cual se explica

Figura 1. El interior de la nueva serie 3000 de Pico. Observa las secciones de entrada protegidas.

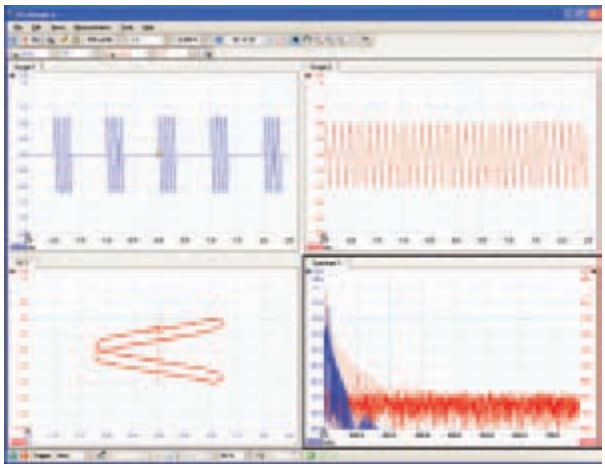


Figura 2. Aquí se han abierto diferentes ventanas en el programa principal.

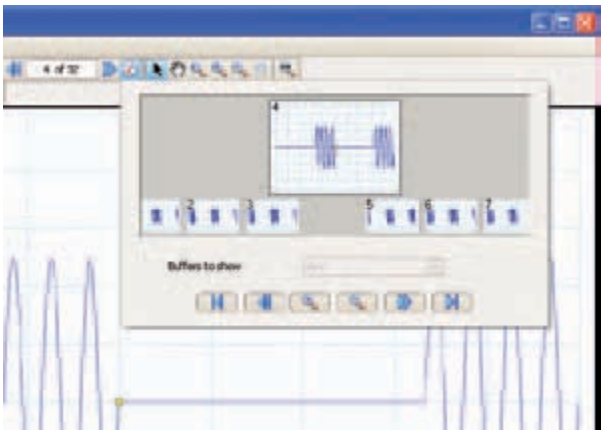
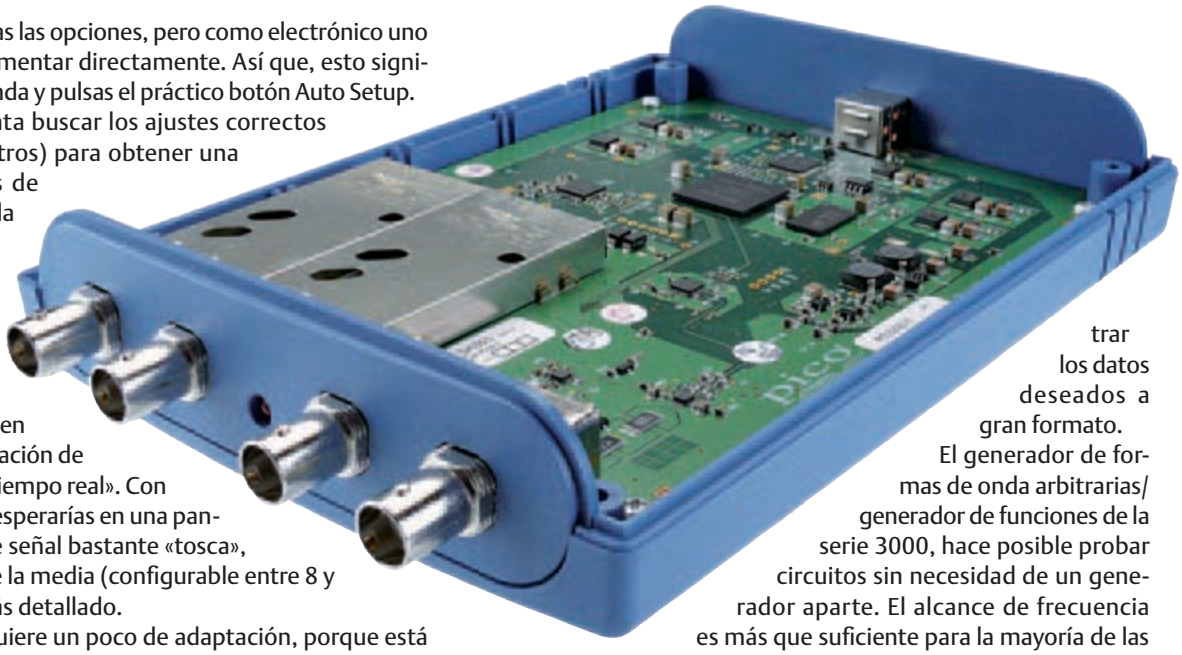


Figura 3. Gracias a la división en bloques que se muestra en el resumen en forma de sellos postales, se puede abarcar rápidamente el contenido del buffer de la memoria.

al detalle el control y todas las opciones, pero como electrónico uno quiere empezar a experimentar directamente. Así que, esto significa que conectas una sonda y pulsas el práctico botón Auto Setup. El propio software intenta buscar los ajustes correctos (X, Y, y disparo, entre otros) para obtener una imagen fija con formas de onda bien definidas. En la mayoría de los casos este botón resulta funcionar correctamente, incluso puedes adaptar algunos ajustes. La imagen reacciona agradablemente rápida a cambios en la señal, esto da una sensación de que estás midiendo en «tiempo real». Con una resolución de 8 bits esperarías en una pantalla grande una onda de señal bastante «tosca», pero gracias al cálculo de la media (configurable entre 8 y 12 bits) todo aparece más detallado.



El diseño del mando requiere un poco de adaptación, porque está estructurado completamente diferente a un osciloscopio normal. Pero después de unos días de trabajo con el software, todo va fluidamente. Pico no ha intentado emular los elementos de control de un osciloscopio convencional y esto es quizás lo mejor, ya que si no, hubiera sido otra vez una combinación de varios botones de osciloscopio con varios menús para las funciones especiales. Ahora estás obligado a pensar de forma completamente diferente. El menú se ha mantenido minimalista, donde sólo las funciones más importantes están disponibles con un clic del ratón. El resto de las cosas se encuentran en la estructura del menú a un nivel más profundo, pero la mayoría de las cosas no tiene pérdida. Puedes conmutar fácilmente entre uno y dos canales de entrada o añadir una ventana nueva con una señal o forma de onda diferente. Las funciones de disparo avanzadas son muy amplias y están documentadas claramente. Hay un menú aparte con una breve explicación y una imagen gráfica para cada opción de disparo. El buffer de memoria se puede recorrer sencillamente con la ayuda de varias pantallas que aparecen en una ventana aparte en formato de sello postal, para después poder mostrar los datos deseados a gran formato.

El generador de formas de onda arbitrarias/generador de funciones de la serie 3000, hace posible probar circuitos sin necesidad de un generador aparte. El alcance de frecuencia es más que suficiente para la mayoría de las aplicaciones (1 MHz). El generador de formas de onda arbitrarias del modelo que hemos probado, ofrece además la posibilidad de diseñar gráficamente diferentes formas de ondas en un tipo de mini editor. Así creé en 5 minutos una ráfaga de seno muy apropiada para evaluar el comportamiento de las ondas oscilantes en filtros de audio y altavoces. Aunque es una señal muy compleja para que un osciloscopio dispare correctamente (sin que los senos «bailen» continuamente), encontré rápidamente la configuración avanzada de disparo donde todo funcionó perfectamente.

Gracias a la alta frecuencia de muestreo de la nueva serie 3000 (500 Msamples/s para un canal, 250 para dos canales) es posible observar con precisión la forma exacta de los flancos de la señal o las tendencias de oscilación incluso de las señales a frecuencias más altas. Esto ofrece, especialmente en combinación con el gran buffer de memoria, muchas posibilidades de análisis. No hemos probado la velocidad máxima de muestreo para señales repetidas continuamente (en el 3206A/B 10 Gsamples/s), pero puede resultar muy útil en algunas situaciones.

Combinación perfecta

La nueva serie 3000 de Pico, ofrece una combinación perfecta de osciloscopio USB con una frecuencia de muestreo bastante alta y un generador de funciones/generador de formas de onda arbitrarias, con un alcance de frecuencia útil. Por 483 euros (IVA excluido) tienes un 3204A con un generador de funciones incorporado y un ancho de banda de 60 MHz. Con esto obtienes un osciloscopio completo con muchas opciones y una pantalla grande (el monitor del ordenador), que no requiere alimentación propia y por eso es muy apto para combinarlo con un portátil. Puedes optar por una versión más cara dependiendo del ancho de banda requerido, del generador de formas de onda arbitrarias y del buffer de memoria. El modelo 3206B que probamos nos gusta mucho para la práctica diaria, incluso tanto que es difícil volver después a trabajar con un osciloscopio convencional. Pico ha construido con esta serie 3000, una serie de instrumentos de medición versátiles con componentes actuales, prestaciones perfectas y una buena relación calidad/precio.

(110268)

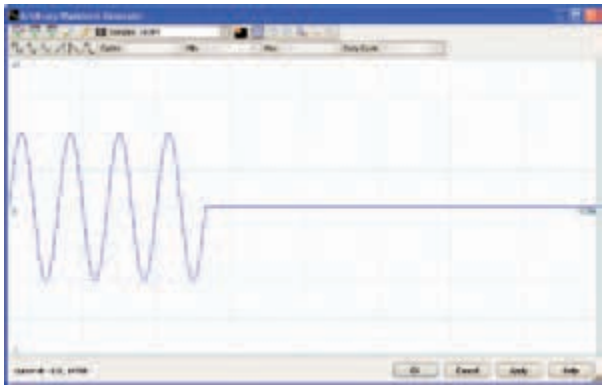


Figura 4. Puedes crear rápidamente una señal especial en la ventana del generador de formas de onda arbitrarias.

Más información:

www.picotech.Com/computer-oscilloscope.html

Seguimiento GPS con ATM18
sin quitar la vista de nuestra placa preferida

Grégory Ester (Francia)

El módulo módem GSM/GPRS GM862-GPS integra un GPS de 20 canales SIRF III que permite localizarnos de forma muy precisa. Así pues, será posible montarlo sobre un "vehículo" ATM18 y jugará así el papel de «trazador» y que será capaz de transmitirnos, por SMS o por correo electrónico, la posición geográfica del móvil en movimiento. Nuestro pequeño(a) amigo(a) nunca más estará sólo(a)...

Nota: Este artículo constituye una continuación lógica a lo publicado el último mes sobre el mismo tema [1]. Sin embargo, podemos realizar los montajes sin recurrir a las explicaciones presentadas en el número precedente. El material utilizado es idéntico: la ATM18 [2] (y su pantalla LCD bifilar [3]) se comunica con el módulo OEM GM862-GPS [4] insertado sobre su placa de prueba [5], equipada con antenas GPS y GSM. El diagrama de cableado (ver **Figura 1**) permite tres funcionamientos diferentes, resumidos en la **Tabla 1**.

El GM862-GPS con un corazón totalmente nuevo

Si acabamos de hacernos con un módulo GM862-GPS y si el *firmware* (programa interno de gestión) no se corresponde con la versión 07.03.402, vamos a tener que hacer una actualización del mismo. Para ello, conectaremos el GM862-GPS al ordenador: en lugar de ser conectadas, respectivamente, a PC2 Y PC3 del ATM18, las señales RXI y TXO serán conectadas sencillamente a las señales TxD (naranja) y RxD (amarillo) de un conversor FTDI USB-SBRIE [6], por ejemplo. Ejecutamos el comando `AT+CGMR$0D`, gracias al terminal Hercules [7], ¿no es la versión 07.03.402 la instalada en nuestro modem? Entonces, he aquí el procedimiento que hay que seguir para actualizarla:

1. Ajusta la velocidad de transmisión de los comandos del módem a 115.200 baudios: `AT+IPR=115200$0D`. Arranque de nuevo el terminal utilizando esta nueva configuración.
2. El firmware, así como la aplicación Xfp 2.13 que permite alojarlo en el módulo GM862-GPS, están disponibles (después de registrarnos) en la página web del constructor [8].
   Hay que señalar que también es posible conseguir estos dos ficheros a través del responsable técnico de Telit para Francia, por medio de una sencilla petición, en francés o inglés, a la dirección siguiente: Khaled.Chtourou@telit.com
3. Alimenta la placa soporte del módulo GM862-GPS. Ejecuta la aplicación Xfp 2.13. Si fuese necesario, apagua el módulo presionando durante un cierto tiempo el botón-pulsador ON/OFF. El LED STAT está ahora apagado. Selecciona el puerto serie utilizado y ajustado a 115.200 baudios, busca en el disco duro (botón *Browse*) el programa. Pulsa sobre el botón *Program*; el mensaje *Linking* aparece, abajo a la izquierda, al mismo tiempo que la barra de avance parpadea (ver **Figura 2**). Pulsa de nuevo sobre el botón-pulsador ON/OFF como lo haríamos con nuestro teléfono móvil para encender el módem. El LED verde STAT queda encendido y la actualización se inicia. El indicador de avance nos indica la progresión (ver **Figura 3**). Una vez hemos actualizado el firmware, el LED STAT se apaga y se presenta el mensaje de la **Figura 4**. Pulsa sobre *OK*, aprieta de nuevo en el botón pulsador ON/OFF de la placa de prueba del módem y ejecuta el comando `AT+CGMR$0D`. ¡La respuesta `07.03.402` aparece sin llamar! A partir de ahora este es el nuevo software que equipa nuestro GM862-GPS.

Restablece la velocidad de ejecución de los comandos del modem a 9.600 baudios por medio del comando `AT+IPR=9600$0D`.

El GPS del GM862-GPS

Aunque un comando AT permite interrogar al módulo para obtener informaciones sobre la geolocalización de éste último, existe también, sobre la placa de prueba, el terminal EMMI_TX que envía el conjunto de las tramas en el formato NMEA0183, independientemente del funcionamiento del conjunto, con la condición de que el GM862-GPS esté alimentado, por supuesto. Una pulsación larga sobre el botón ON/OFF nos permite poner en funcionamiento el GM862-GPS; el LED STAT parpadea a intervalo de 1 s, indicando que el módulo no está aún registrado en la red, lo que no le

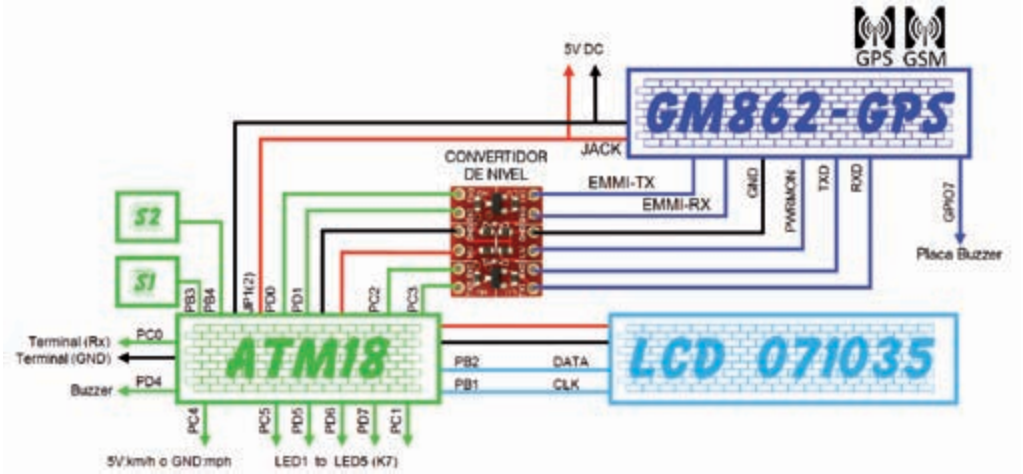


Figura 1. Diagrama de cableado.

Tabla 1. Tres microprogramas a elegir	
Fichero	**Función**
110267-I_GM862-GPS_ATM18_P3.bas	Visualización sencilla en la pantalla LCD de las informaciones útiles provenientes de las tramas NMEA
110267-I_GM862-GPS_ATM18_P4.bas	Geolocalización de un vehículo por SMS o correo electrónico
110267-I_GM862-GPS_ATM18_P5.bas	Identificación de las BTS (<LAC>+<CI>), una tras otra + recuperación de la calidad de la señal + posición geográfica

impide enviar las tramas NMEA. Por defecto, las tramas GGA, GSA, GSV, RMC son enviadas a 4.800 baudios, una detrás de otra, sobre el terminal EMMI_TX.
En un primer momento, nuestro primer microprograma (110267-I_GM862-GPS_ATM18_P3.bas) va a permitir explotar las tramas GGA y RMC.
La trama GGA nos permitirá recuperar la hora UTC, la latitud, la longitud, el indicador de validez de los datos, el número de satélites perseguidos y la altitud.
En cuanto a la trama RMC, nos permitirá recuperar la velocidad y la fecha. Alrededor de cada segundo, estos ocho datos son mostrados en nuestra pantalla LCD bifilar de 4 líneas de 20 caracteres. Una vez hemos cargado el microprograma, si los datos son válidos y el número de satélites utilizados para calcularlos es superior o igual a cuatro, los resultados se mostrarán de acuerdo a la **Figura 5**. Es posible visualizar la pantalla de la **Figura 6** por medio de una pulsación sobre el botón S2. Si el terminal PC4 está conectado a masa, la velocidad se presenta en la tercera línea, en millas por hora. Si PC4 está conectado a 5 V, la velocidad se muestra en kilómetros por hora (km/h) sobre la misma línea (ver **Figura 7**).
Por defecto, la visualización de la hora universal está ajustada a UTC+1 por programa.

La ATM18 delatora

Nuestra misión, si la aceptamos, consiste en enviar un correo electrónico por el puerto 25 de un servidor SMTP utilizando el módulo GM862-GPS, ¡como es lógico!
El uso del correo electrónico es un servicio accesible gracias al GPRS. En ciertos operadores nos será necesario pedir una activación de este servicio. Una vez activado el servicio (**Figura 8**), ya nos será posible abrir una sesión de internet y, por consiguiente, enviar un correo electrónico por GPRS. Des-

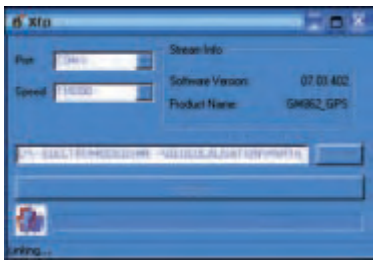


Figura 2. Xfp *Ready*.

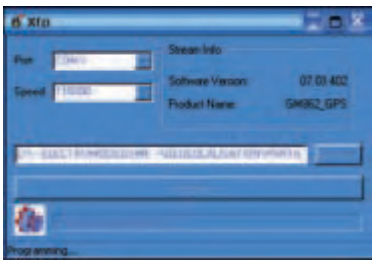


Figura 3. ¡Xfp *GO*!

Figura 4. ¡Xfp *Done*!

Figura 5. Buenos días, son las 20h 43 UTC+1.

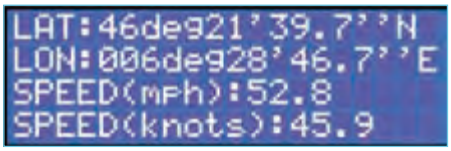


Figura 6. *¿Where are you?* (¿Dónde esta?)

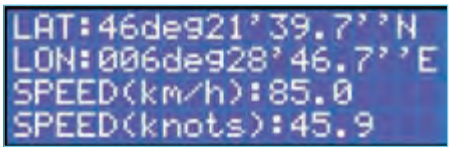


Figura 7. ¿Où es-tu? (¿Dónde estás?)

pués, el microprograma **110267-I_GM862-GPS_ATM18_P4.bas** se encargará de ello, en nuestro lugar, con una simple petición.
Antes de proseguir, la **Tabla 2** nos ayudará a reunir todos los parámetros y las informaciones necesarias para el envío de un correo electrónico utilizando el GM862-GPS. En nuestro caso de escuela, descrito más adelante, somos el remitente (*Sender*), es decir, el que desea enviar un correo electrónico (*e-mail*) a un destinatario (*Receiver*). Después de haber introducido el código PIN, todas estas informaciones nos van a permitir crear un "contexto" GPRS, gracias a los comandos que siguen (los "OK" son las respuestas del módulo):

```
AT+CPIN=7453<CR>
OK
AT+CGDCONT=1,»IP»,»internet68»
<CR>
OK
AT#esmtp=»smtp.mail.yahoo.
fr»<CR>
OK
AT#euser=»gpstracker74»<CR>
OK
```

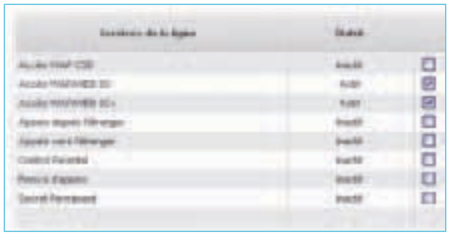


Figura 8. GPRS activo.

```
AT#epassw=»258369»<CR>
OK
AT#eaddr=»gpstracker74@yahoo.
fr»<CR>
OK
AT#esav<CR>
OK
```

Una vez creado el contexto GPRS y salvado, nos basta con activarlo con el comando AT:

```
AT#SGACT=1,1<CR>
#SGACT: 10.189.67.153
OK
```

Este comando nos proporciona, como respuesta una dirección IP que identifica al módulo GM862-GPS sobre la red. En esta fase, nuestro módem está preparado para enviar correos electrónicos. Es lo que vamos a hacer utilizando el comando siguiente:

```
AT#EMAILD="contact@adelek.
fr","TEST",0<CR>
```

El módem responde con ">" y aquí podemos escribir nuestro mensaje. Validamos nuestro mensaje con <Ctrl>-Z:

```
> Bonjour, ceci est un message
de test.<Ctrl>-Z
OK

(> Buenos días, este es un men-
saje de prueba.<Ctrl>-Z
OK)
```

Antes de compilar el programa **110267-I_GM862-GPS_ATM18_P4.bas**, habrá que modificar ciertas constantes en función de las informaciones utilizadas un poco antes.

Tabla 2. Parámetros e informaciones a recabar antes del ejercicio de «envío de un correo por GPRS»

Parámetros	Datos (ejemplo)	Nuestros datos
Servidor SMTP	smtp.mail.yahoo.fr	
Mensaje de remitente (Sender)	gpstracker74@yahoo.fr	
Mensaje del destinatario (Receiver)	contact@adelek.fr	
Objeto del mensaje	TEST	
Cuerpo del mensaje	Test GPRS feature, hello World !	
Nombre del punto de acceso (APN, dependiente de nuestro operador)	internet68	
Login de la cuenta de mensajería del remitente (USERID)	gpstracker74	
Contraseña de la cuenta de mensajería del remitente (PASSWORD)	258369	

Figura 9. Geolocalización por SMS.

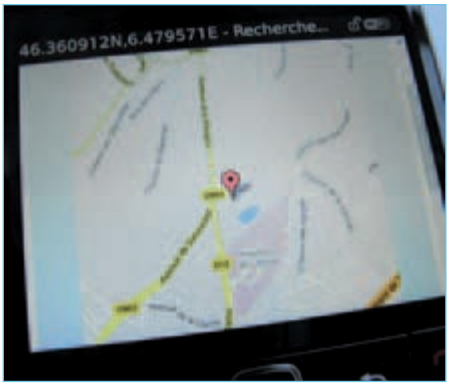


Figura 10. ¿Dónde está mi placa ATM18?

Las cuatro contraseñas, cuya función está descrita justo a continuación, deben contener siete caracteres (letras o cifras) sin acentos, ver el **Listado 1**.

Una vez compilado el programa y cargado en memoria, tenemos aquí cómo reacciona nuestro nuevo juguete: el conjunto del material será colocado en un vehículo al que deseamos seguir en su desplazamiento. Nuestra compañera abandona el domicilio al volante del mencionado vehículo y nosotros permanecemos en casa. Si queremos interrogar al circuito hay dos soluciones.

La primera consiste en enviar un SMS que, como contenido, tenga la contraseña que hemos definido previamente (T090471, en nuestro ejemplo). Un pequeño tiempo de espera y recibimos como respuesta un SMS automáticamente (ver **Figura 9**) donde se nos da información del par latitud/longitud, de un enlace de Internet con el formato *google-maps*, la fecha y la hora, la velocidad, la altitud, así como el número de satélites que se han usado para calcular estos valores. Si tenemos acceso a Internet desde nuestro móvil, pulsamos sobre el enlace *google-maps* y se muestra el mapa (en la URL el parámetro "t=m" nos mostrara el dibujo de un mapa, el zoom está fijado en 10 por "z=10"), la pequeña bandera clásica indica la posición del vehículo en cuestión (**Figura 10**).

La segunda solución nos permite recibir, en respuesta al envío del SMS "E090471", las mismas informaciones pero, esta vez, por correo electrónico, a la dirección "Email1" (**Figura 11**), es decir, en nuestro caso, contact@adelek.fr. El objeto del correo recibido contiene la hora a la que se ha tomado el conjunto de las informaciones transmitidas.

Un segundo usuario también puede interrogar al montaje de la misma manera, enviando su contraseña (T180676 o E180676, en este ejemplo). La respuesta le será enviada, bien por SMS al teléfono móvil "Phone2" o bien por correo electrónico a la dirección "Email2".

Hay que señalar aquí que la pantalla LCD bifilar no es utilizada por este programa. La visualización de las informaciones se realiza por medio de cinco LED cableadas según el diagrama de la **Figura 1**.

Listado 1

```
Const Decal = 1      'UTC + 1
'
Const Apn = «internet68»    'APN
Const Esmtp = «smtp.mail.yahoo.fr»       'EMAIL SENDER SMTP
Const Euser = «gpstracker74»             'EMAIL SENDER LOGIN
Const Epassw = «258369»                  'EMAIL SENDER PASSWORD
Const Eaddr = «gpstracker74@yahoo.fr»    'EMAIL SENDER NAME
'
Const Passw1t = «T090471»                'PASSWORD USER1
Const Passw1e = «E090471»
Const Phone1 = «0682834725»              'PHONE USER1
Const Email1 = «contact@adelek.fr»       'EMAIL USER1
'
Const Passw2t = «T180676»                'PASSWORD USER2
Const Passw2e = «E180676»
Const Phone2 = «06XXXXXXXX»              'PHONE USER2
Const Email2 = «stephanie.b@free.fr»     'EMAIL USER2
'
Const Code_pin = «7453»                  'SIM PIN
```

¡BTS! ¡Tenemos los medios de hacerles hablar!

Un teléfono móvil está relacionado con la red GSM gracias a una estación base BTS (*Base Transceiver Station*). A medida que nos desplazamos con nuestro teléfono, navegamos de celda en celda sin perder nunca la conexión de nuestra conversación, ¿no es así? [...] fin [...] así! Cada celda posee un identificador único, una manera de situarnos, más o menos, aproximadamente. La identificación completa del emisor-receptor (BTS) del cual dependemos es una pareja de valores LAC (de *Local Area Code*) y CI (de *Cellule Identity*), que nos es enviada con cada cambio de celda por la estación de base.

La ATM18 podrá ser configurada de manera que recupere automáticamente el par de datos zona+celda a medida que cambiamos de posición. Así, podremos completar nuestra propia base de datos que contendrá los números de las estaciones de base correspondientes a nuestra zona geográfica, por ejemplo. Por defecto, la recepción de este par de valores no está activada, es el comando `AT+CREG=2` (*enable network registration unsolicited result code with network cell identification data*) el que permite autorizar esta función.

El programa **110267-I_GM862-GPS_ATM18_P5.bas**, una vez compilado y cargado en memoria, nos va a permitir recuperar, uno tras otro, durante un trayecto en coche, por ejemplo, la identidad de cada estación base utilizada. A cada cambio de celda, el programa le envía el conjunto de las informaciones, a 9.600 baudios, sobre PC0 (ver **Figura 1**), hacia el ordenador sobre el que se está ejecutando

Tabla 3. Funciones de los LED para el microprograma 110267-I_GM862-GPS_ATM18_P4.bas.	
LED	**Función**
1	El número de satélites seguidos es inferior a 4 o los datos provenientes de la trama GGA no son válidos o recuperación en curso de los datos provenientes de tramas válidas
2	Los comandos de configuración son enviados al GM862-GPS o el módulo no capta la red
3	En espera de un SMS
4	Envío de un SMS como respuesta a una petición de posicionamiento recibida por SMS
5	Envío de un correo electrónico como respuesta a una petición de posicionamiento recibida por SMS

Tabla 4. Funciones de los LED para el microprograma 110267-I_GM862-GPS_ATM18_P5.bas.	
LED	**Función**
1	El número de satélites seguidos es inferior a 4 o los datos provenientes de la trama GGA no son válidos o recuperación en curso de los datos provenientes de tramas válidas
2	Los comandos de configuración son enviados al GM862-GPS o el módulo no capta la red
3	En espera de sintonizar una nueva estación base (BTS)
4	Recuperación del par LAC + CI que identifica la nueva estación base (BTS), de la calidad de la señal en ese instante y envío hacia el terminal de estas informaciones mas las correspondientes al posicionamiento

el programa terminal Hercules, configurado como *Logger* (para permitirnos registrar directamente los datos en un fichero). Los datos están organizados de la siguiente manera:

Figura 11. Geolocalización por correo electrónico.

Figura 12. 40 km, 27 BTS.

```
----------
Start with BTS : +CREG: 2,1,»296A»,»4437»
Signal Quality : +CSQ: 30,0
13/03/2011
16h16m15.000s
SAT:05 ALT:462.1M
LAT:46deg21’40.9’’N
LON:006deg28’44.5’’E
http://maps.google.com/maps?q=46.361347N,6.479033E&t=m&z=10
----------
----------
BTS : +CREG: 1,»296A»,»28D1»
Signal Quality : +CSQ: 21,0
13/03/2011
16h17m02.000s
SAT:08 ALT:464.4M
LAT:46deg21’40.8’’N
LON:006deg28’44.5’’E
http://maps.google.com/maps?q=46.361320N,6.479023E&t=m&z=10
----------
----------
BTS : +CREG: 1,»296A»,»A2B3»
Signal Quality : +CSQ: 21,0
13/03/2011
16h21m52.000s
SAT:08 ALT:448.7M
LAT:46deg21’33.9’’N
LON:006deg27’48.8’’E
http://maps.google.com/maps?q=46.359409N,6.463560E&t=m&z=10
----------
…
```

El programa no utiliza la pantalla LCD bifilar, por lo que cuatro LED (**Tabla 4**) nos indicaran las informaciones relativas al funcionamiento del montaje. El montaje ha sido probado sobre una distancia de 40 km (ver **Figura 12**). En este trayecto

Enlaces en internet

[1] www.elektor.fr/110139
[2] www.elektor.fr/atm18
[3] www.elektor.fr/071035
[4] www.telit.com/en/products/gsm-gprs.php?p_ac=show&p=7
[5] www.sparkfun.com/products/281
[6] www.elektor.fr/magazines/2008/juin/cable-usb-seriel-ttl.500289.lynkx
[7] www.hw-group.com/products/hercules/index_en.html
[8] www.telit.com/en/products/download-zone.php
[9] www.elektor.fr/110267

El módulo GM862-GPS, así como las antenas GPS y GSM, la placa de prueba y la placa de conversión de nivel, son vendidos por Lextronic, www.lextronic.fr

Tabla 5. Identificación de diez (entre 27, ver Figura 12) estaciones base recuperadas, por agrupación de tamaño medio, sobre una distancia de 40 km.

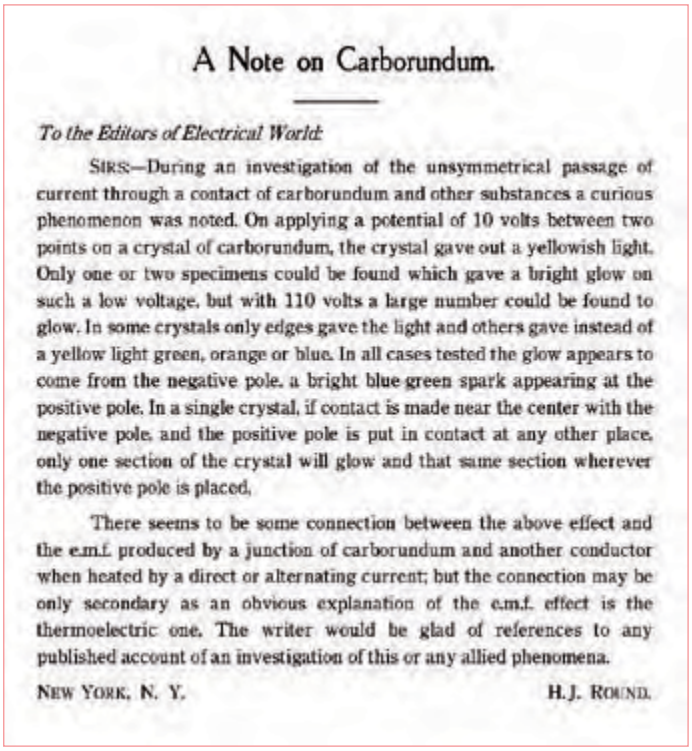
LAC	CI
296A	1D43
296A	B93B
296A	F11D
296A	94CC
296A	B93B
296A	94CC
296A	B93B
296A	922B
296A	94CC
296A	C2B8

se pudieron identificar 27 estaciones base (**Tabla 5**).

¿Os ha servido este artículo? Dilo con palabras a tellme@adelek.fr

(110267)

Luz sólida: la Famosa Historia



A Note on Carborundum.

To the Editors of Electrical World:

SIRS:—During an investigation of the unsymmetrical passage of current through a contact of carborundum and other substances a curious phenomenon was noted. On applying a potential of 10 volts between two points on a crystal of carborundum, the crystal gave out a yellowish light. Only one or two specimens could be found which gave a bright glow on such a low voltage, but with 110 volts a large number could be found to glow. In some crystals only edges gave the light and others gave instead of a yellow light green, orange or blue. In all cases tested the glow appears to come from the negative pole, a bright blue green spark appearing at the positive pole. In a single crystal, if contact is made near the center with the negative pole, and the positive pole is put in contact at any other place, only one section of the crystal will glow and that same section wherever the positive pole is placed.

There seems to be some connection between the above effect and the e.m.f. produced by a junction of carborundum and another conductor when heated by a direct or alternating current; but the connection may be only secondary as an obvious explanation of the e.m.f. effect is the thermoelectric one. The writer would be glad of references to any published account of an investigation of this or any allied phenomena.

NEW YORK, N. Y. H. J. ROUND.

Figura 1. H.J. El informe de Round en Mundo Eléctrico, 1907.

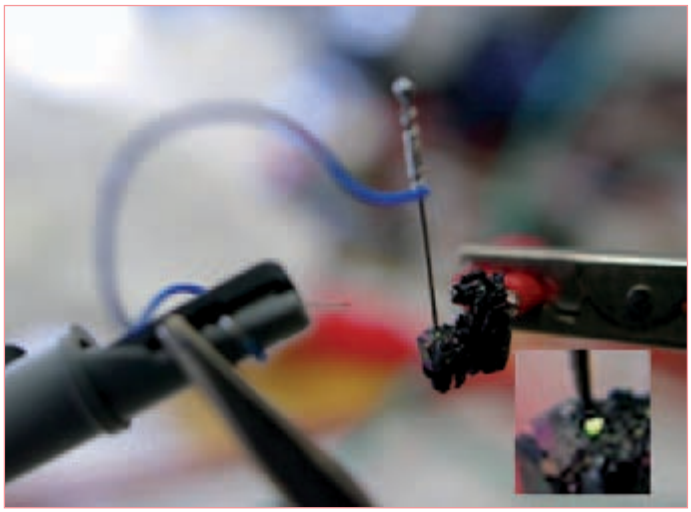


Figura 2. Michael Lippert ha reproducido los experimentos de Round de 1907 que reveló el efecto que es la base del fotodiodo (LED). Aquí una aguja negativamente cargada se pone en contacto con un cristal de carburo de silicio positivamente cargado. Con un voltaje de 9 V y una corriente de 30 mA un brillo verde puede ser observado al punto de contacto entre aguja y alfiler. (Wikimedia Commons)

Andrew Emmerson (Reino Unido)

¿Quién inventó el diodo emisor de luz o LED? ¡Nadie! Replanteemos la pregunta de nuevo; ¿Quién fue el primer descubridor de la emisión de luz de sustancias semiconductoras? La respuesta es el asistente de investigación de Marconi, Henry (H. J.) Round, en 1907. Pero Round estaba ocupado con otro trabajo y dejó que otros elaboraran - o descubrieran por separado - sus conclusiones.

Su informe (ver **Figura 1**), realizado mientras preparaba los detectores de pelo del gato (diodos) para experimentos de radio, notó que un potencial de corriente continua aplicado entre dos puntos sobre un cristal del carborundo (carburo de silicio, SiC) provocó que el cristal produjera luz. Él propuso un posible vínculo entre el voltaje a través de la unión de carborundo y la emisión de luz.

Al principio el descubrimiento de Round tuvo poco interés, aunque 102 años más tarde esto inspiró al experimentador alemán Michael Lippert a recrear estos ensayos (ver **Figura 2**). Podemos ver más fotografías y detalles prácticos en su blog [1]. A diferencia de los que 'no intentan proezas en casa', el experimento de Michael es aparentemente sencillo. Todo lo que necesitamos es una fuente de energía de 9 V, un alfiler (formando el detector de pelo de gato), un cristal de carborundo (que Michael compró barato en eBay) y un cuarto oscuro. Si movemos los alfileres despacio a través de la superficie del cristal y deberíamos ser premiados con un brillo diminuto de luz de colores al punto de contacto. Según el punto y la cantidad de la corriente fluida que hayamos elegido, nuestor diodo LED casero puede aparecer en naranja, amarillo o verde. Si Dios quiere puede ser completamente brillante, declara Michael.

Las grabaciones de emisión de luz de semiconductores aparecen a principios de los años 1920, cuando un técnico de radio ruso conocido con el nombre de Oleg Losev (ver **Figura 3**), también escrito como Losov o Lossev y pronunciado como Olyeg Lossov, hizo el mismo descubrimiento que Round hizo antes. En otras palabras, él notó que los diodos de cristal usados en receptores de radio emitían luz cuando la corriente pasaba por ellos. A diferencia de Round, él investigó en profundidad el objeto de la electroluminescencia en el Instituto Physico-técnico en Leningrado, publicando varios artículos científicos que hablaban de las características de voltaje corriente de diodos de SiC y su relación con la emisión de luz. Uno de estos artículos, 'El detector de carborundo luminoso y el efecto de descubrimiento y las oscilaciones con los cristales, de la Revista (noviembre 1928) periódica británica *Philopsophical* tenía 21 páginas.

Mientras que Round vió la electroluminiscencia sólo como una curiosidad, Losev no tenía ninguna duda sobre el significado práctico de su descubrimiento. Las fuentes de la luz no-vacías en miniatura funcionaron en baja tensión (menos de 10 V) y en la muy alta velocidad podían realizar funciones que las bombillas normales no podían y su patente de 1927 para un 'relevo ligero' predestinó su uso

del Diodo LED

Figura 3. Oleg Losev, el técnico de radio ruso que notó que los diodos usados en receptores de radio emitían luz cuando corriente pasaba por ellos. (Wikimedia Commons)

Класс 21 а, 27

№ 12191

ПАТЕНТ НА ИЗОБРЕТЕНИЕ

ОПИСАНИЕ

светового реле.

Предлагаемое изобретение использует общеизвестное явление свечения в карборундовом детекторе и состоит в том, что в световом реле для быстропишущего телеграфного или телефонного приема, передачи изображений на расстояние и других целей, в качестве модулируемого электрическим током источника света, применяется свечение в точке контакта карборундового детектора, включенного непосредственно в цепь модулирующего тока.

На чертеже фиг. 1 изображает схему предлагаемого светового реле и фиг. 2 — схему устройства для фотографической записи сигналов с применением светового реле.

К зажимам *A* источника тока сигналов, подлежащих записи, через потенциометр *P* включается светящийся карборундовый детектор *D*, в цепь которого включена батарея *B*, дающая дополнительное постоянное напряжение. Для наложения его на напряжение тока сигналов и усиления действия реле; путем регулировки этой батареи создаются наивыгоднейшие условия работы детектора *D*. Оптическая система *L* предназначена направлять световой поток, излучаемый карборундовым детектором, на движущуюся фотографическую пластинку *F*, на которой производится запись изменений этого потока. Детектор *D*, оптическая система *L* и пластинка *F* заключены в светонепроницаемую камеру. Примерное выполнение светового реле показано на чертеже 2, где *E* — приемник-усилитель высокой частоты, *T* — автотрансформатор высокой частоты, а остальная часть схемы вполне аналогична только-что описанной.

Предмет патента.

1. Световое реле для быстропишущего телеграфного или телефонного приема, передачи изображений на расстояние и для других целей, характеризующееся применением, в качестве модулируемого электрическим током источника света, свечения в точке контакта карборундового детектора общеизвестного устройства, каковой детектор включен непосредственно в цепь модулирующего тока.

2. Видоизменение охарактеризованного в п. 1 светового реле, отличающееся тем, что последовательно с указанным детектором *D* включен источник дополнительного напряжения постоянного тока *B* (фиг. 1 и 2) с целью усиления действия реле.

Figura 4. La patente de la URSS publicó a Oleg Losev en 1927 para su sistema el 'relevo ligero' (cortesía PatentsFromRU.com).

"para la comunicación telegráfica rápida, comunicación telefónica y transmisión de imágenes y otras aplicaciones cuando se utiliza un punto de contacto de luminescencia ligero como la fuente de la luz relacionada directamente con un recorrido de la corriente modulada" - en otras palabras los sistemas optoelectrónicos que hacen las comunicaciones modernas posibles. Desgraciadamente sus prometerdoras investigaciones se terminaron en 1942 cuando murió de hambre en el Sitio de Leningrado, a la edad de tan sólo 39 años.

¿Merecen por tanto Round y Losev más crédito en sus descubrimientos con semiconductores de emisión de luz? Probablemente no, lo llevaron a un callejón sin salida (¡no es un juego de palabras!). Como Wikipedia explica, el carburo de silicio es un material ineficaz para fotodiodos. Por esto, ni el trabajo pionero de Round ni el de Losev, incluso si hubieran continuado, probablemente conduciría al éxito comercial.

(110021)

Enlaces en Internet

[1] http://www.dlip.de/?p=99 Michael Lippert construye un LED de de SiC "Hágalo Usted Mismo".

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 redaccion@elektor.es.