Nº 320 - Octubre 2014 - 6.50 €

Nueva Electrónica 3.0

Revista mensual de diseño electrónico, actualidad científica y novedad tecnológica

ISSN 0212-6249
00320
9 770212 624004

www.nuevaelectronica.com

Lo mejor de la tecnología
para tu hobby preferido

velleman Partnership FUTURA ELETTRONICA

IMPRESORA 3D en kit
cod. K8200

¡Haz autónoma la impresora!

Controlador autónomo para impresora 3D.
cod. VM8201

¡Imprime su contenedor con la 3D!

Shield por Arduino:

RGB SHIELD	AUDIO SHIELD	MOTOR SHIELD	ETHERNET SHIELD	IN/OUT SHIELD
				
kit cod. KA01	kit cod. KA02	kit cod. KA03	kit cod. KA04	kit cod. KA05
montado cod. VMA01	montado cod. VMA02	montado cod. VMA03	montado cod. VMA04	montado cod. VMA05

PRODUCTOS DISPONIBLES EN TODOS LOS DISTRIBUIDORES
VELLEMAN DE ESPAÑA · *WWW.VELLEMAN.EU*

Nueva Electrónica

www.nuevaelectronica.com
Volumen 31 – Época 2 - Número 320
Octubre de 2.014
Depósito Legal: M-18437-1983
ISSN 0212-6249

Director
Eduardo Corral Muñoz
ecorral@nuevaelectronica.com

Redacción
Miguel Alberte, Ernesto Corral, Gabriele Daghetta, Paolo Gaspari, Boris Landoni, ...
redaccion@nuevaelectronica.com

Edita
Board and Book, s.l.
Riaño, 3 – 28042 – Madrid, España
Teléfono: +34 91 187 16 19
www.boardandbook.com
Info@boardandbook.com

Contacto
Revista Nueva Electrónica
Apartado de Correos 62048
28080 – Madrid, España
Teléfono: +34 91 187 16 19
www.nuevaelectronica.com
revista@nuevaelectronica.com

Publicidad y marketing
publicidad@nuevaelectronica.com

Suscripciones
suscripciones@nuevaelectronica.com
Nueva Electrónica se publica 12 veces al año.
Consulta las modalidades de suscripción en:
www.nuevaelectronica.com

Imprime
Grupo Cibeles

CONTENIDOS

ARISTON ELECTRONICA, S.A.

módulos y balizas

energía solar autónoma

www. ariston.es

JH001
Señalización para la construcción
Decoración de plazas, parques y patios

JH002
Colocación en cualquier superficie
Circunvalaciones, intersecciones, autopistas y autovías

JH003
Especialmente para laterales o márgenes de autopistas, autovías, señalización de aceras y senderos

JH004
Por sus características puede ser colocado en columnas de perquings o muros.

JH005
Señalización de medianas y arcenes de autopistas, intersecciones y stops, carreteras secundarias.

JH006
Decora al tiempo que ilumina plazas, parques, patios y embelece veredas.

JH007
Para iluminar y realzar en colores, jardines, parques, patios, muros, veredas.

JH008
Diseñado especialmente para la demarcación y señalización de cualquier espacio fluvial y marítimo, puertos deportivos, lagos, canales, piscinas.

JH009
Decora y señala rutas de plazas, muros y senderos

JH016
Especialmente para laterales o márgenes de autopistas, autovías, señalización de aceras y senderos (plana)

JH018
Señalización para la construcción y señalización del mar (faros)

JH019
Decora y señala rutas de plazas, parques, muros y senderos (forma de trébol)

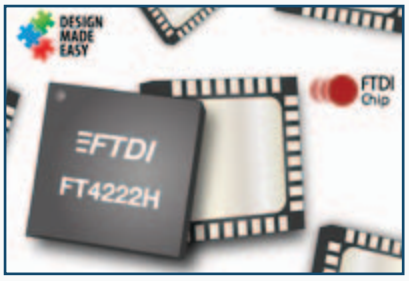
JH022
Luz para la señalización de peligro

- Módulos integrados estancos
- Expectativa de vida hasta 20 años
- Anti-vandálico
- Resistentes al agua
- No necesitan instalación eléctrica
- ISO 9001

Chip puente avanzado, con numerosas E/S y un consumo eficiente entre USB 2.0 y SPI/I2C

Ofrece soporte a múltiples líneas de datos y una amplia variedad de configuraciones diferentes para maximizar la flexibilidad de diseño

FTDI Chip ha sido considerado durante mucho tiempo un sinónimo de innovación en USB y la compañía continúa presentando nuevos productos semiconductores que ayudan a los ingenieros a implementar mejores diseños. El FT4222H es la incorporación más reciente a su creciente oferta USB. Este dispositivo, conforme al estándar de alta velocidad USB 2.0, es una solución de puenteo monochip con capacidades de interface I2C y SPI multicanal.
Por lo que respecta a SPI, el dispositivo sirve como controlador de interface SPI Maestro/Esclavo y ofrece soporte a los 4 modos de SPI (0, 1, 2, 3). Permite la transferencia de datos de anchura sencilla, doble y cuádruple, por lo es posible alcanzar una velocidad total de transmisión de los datos de hasta 27Mbit/s. El controlador de interface I2C Maestro/Esclavo configurable integrado cumple totalmente las especificaciones I2C v2.1 y v3.0. Puede trabajar en modo estándar de 100kbit/s (SM), modo rápido de 400kbit/s (FM), modo rápido plus de 1Mbit/s (FM+) y modo de alta velocidad de 3,4Mbit/s (HS). Las E/S configurables de aplicación general (general purpose IOs, GPIO) se pueden controlar fácilmente mediante aplicaciones de software a través del bus USB. El interface del USB 2.0 consume una corriente muy baja tanto en modo activo (75mA típica) como en modo suspendido (375µA típica).
Al igual que otros controladores de dispositivos USB de FTDI Chip, el CI se encarga íntegramente del protocolo USB, por lo que estas tareas no desperdician valiosos recursos en el microcontrolador del sistema, lo cual podría comprometer las prestaciones previstas. La memoria OTP integrada en el FT4222H permite almacenar la identificación del suministrador de USB (VID), la identificación del producto (PID), el número de referencia del dispositivo y la descripción del producto, así como otros datos del suministrador. La función de detección del cargador de batería dota a los periféricos USB que incorporan este CI de la capacidad de detectar la conexión a un puerto de carga dedicado (dedicated charging port, DCP) y gracias a ello obtienen una carga más rápida. Los drivers propietarios y sin royalties de FTDI Chip para el sistema operativo Windows permite que los ingenieros logren, en la mayoría de las circunstancias, evitar el inconveniente de tener que

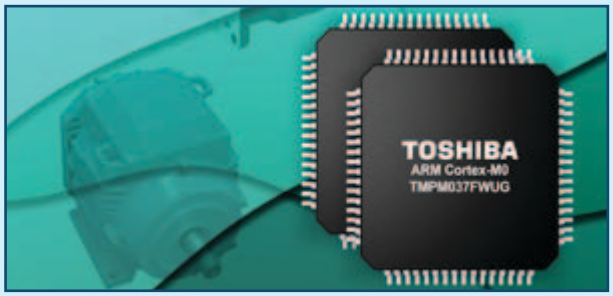


desarrollar sus propios drivers.

"El historial de FTDI Chip en USB es único. La incorporación de un chip puente SPI cuádruple a nuestro catálogo resulta muy ventajosa y esperamos una excelente respuesta. Este dispositivo proporciona más líneas de datos a los ingenieros, por lo que se pueden transferir grandes cantidades de datos utilizando menos ciclos de reloj", explica Satyajit Sarma, Director de la Línea de Productos USB en FTDI Chip.
El FT4222H se suministra en un encapsulado compacto QFN de 32 patillas que no contiene plomo y con un rango de temperaturas de trabajo entre -40°C y 85°C por lo que resulta indicado para aplicaciones industriales exigentes..

www.ftdichip.com

Toshiba presenta un ARM Cortex-M0 multi-función con número de pines reducido



Toshiba Electronics Europe (TEE) ha introducido un nuevo microncontrolador a su serie TX00 de microcontroladores basados en el núcleo ARM® Cortex®-M0 (MCU). El microcontrolador TMPM037FWUG está diseñado para aplicaciones de control de motores en equipos tales como impresoras multifuncionales (MFP), impresoras, electrónica de consumo, equipos digitales y equipos de automatización de fábrica.
El desarrollo de aplicaciones de control de motores altamente sofisticados en los dispositivos modernos requiere varias funciones, incluyendo múltiples canales de comunicación que pueden comunicarse con los dispositivos principales de dispositivos de control y periféricos, interfaces para la lectura de los valores numéricos de los sensores, y una función de temporizador que pueda dar pulsos para controlar un motor, tal como un motor paso a paso o un motor DC.
El TMPM037FWUG integra un convertidor A/D 10 bit de 8 canales y un temporizador 16 bit de 10 canales con un generador de impulsos programable (PPG).
El microcontrolador también integra una interfaz en serie de 6 canales (5 canales SIO/UART, 1 canal I2C), que elimina la necesidad de ICs de extensión interfaz y también contribuye a reducir el coste de fabricación. También tiene una función de procesamiento de banda de bits que soporta el acceso y control de los bits específicos. Esto aumenta la eficiencia de la manipulación de bits y permite la optimización de la memoria flash de pequeña capacidad del producto, que comprende 128 Kbytes Flash y 16 KB de SRAM.

www.toshiba-components.com

NI anuncia un sistema de visión artificial robusto y compacto para cámaras USB3

NI CVS-1459RT delivers quad-core Intel Atom Processor, USB 3.0 camera ports and FPGA-enabled I/O

National Instruments ha presentado una solución compacta para las aplicaciones de visión artificial de alta velocidad. El NI CVS-1459RT es un sistema de visión artificial pequeño y robusto con un procesador Intel Atom de cuatro núcleos y dos puertos USB 3.0 dedicado a las cámaras USB3.

"El sistema compacto de visión artificial (NI Compact Vision System) es mi solución lista para usar en las aplicaciones de visión artificial donde la fiabilidad y el tiempo de actividad son clave", dijo Robert Eastlund, vicepresidente de ventas de Graftek Imaging Inc. "Ahora puedo aprovechar la fácil conectividad y el alto rendimiento del bus USB3 al mismo tiempo que aprovecho las nuevas características de procesamiento de alto rendimiento e integración de HMI. El NI CVS-1459RT hace posibles las soluciones de visión

artificial industrial de alta resolución y elevada velocidad".

El NI CVS-1459RT se programa con el software de diseño de sistemas LabVIEW o con Vision Builder para AI (Automated Inspection). Los ingenieros tienen la opción de usar LabVIEW FPGA para personalizar aún más las E/S de la FPGA y sincronizar estrechamente los resultados de la inspección mediante visión artificial con otras partes de los sistemas industriales, tales como codificadores y sensores de proximidad.

El NI CVS-1459RT se basa en la arquitectura RIO (Reconfigurable I/O) de LabVIEW, una parte integral de la plataforma de diseño gráfico de sistemas de NI. Un método moderno para el diseño, creación de prototipos y despliegue de sistemas de monitorización y control embebidos. El diseño gráfico de sistemas combina el entorno de programación de LabVIEW con el hardware disponible en el comercio para simplificar drásticamente el desarrollo, de manera que los ingenieros pueden combinar potentes herramientas de visión artificial, E/S, comunicación industrial, registro de datos e interfaces hombre-máquina (HMI) en un único entorno.

www.ni.com

Chroma presenta la serie 62000P de fuentes de alimentación programables

Las fuentes de alimentación CC programables de la serie 62000P de Chroma, distribuidas en España por Instrumentos de Medida s.l., ofrecen muchas ventajas únicas para pruebas e integración de sistemas automáticos. Estas incluyen una curva de funcionamiento a potencia constante, lecturas precisas de los valores aplicados de voltaje y corriente, señales de trigger de salida así como la habilidad de crear formas de onda transitorias CC complejas para ensayo del comportamiento de dispositivos ante picos, caídas y otras desviaciones de voltaje. Diseñadas para pruebas automáticas de convertidores DC-DC y productos similares, la familia 62000P es en un nuevo estándar de fuentes programables de alta precisión. La serie 62000P incluye 12 modelos diferentes con potencias desde 600W a 5000W, con corrientes hasta 120A y voltajes hasta 600V. Gracias a su función de curva de operación a potencia constante, un único instrumento puede proporcionar alto voltaje/corriente Y baja tensión/corriente, reduciendo así el número de fuentes necesarias para aplicaciones automáticas típicas.

Las fuentes 62000P también incluyen capacidad de lectura precisa de los valores de salida de voltaje y corriente de 16 bit. Esto elimina de los sistemas los shunts y multiplexores necesarios en el pasado para obtener lecturas precisas de los parámetros de entrada del dispositivo bajo test. También incorporan puertos de Entrada/Salida con señales TTL de 8 bit para activación y parada de la señal de salida, señal de salida fallidasí como señales de disparo para medidas de tiempo y sincronización.

Otra función única de la serie 62000P es la capacidad de crear formas de onda de transitorios CC complejas, permitiendo probar dispositivos ante caídas de voltaje CC, picos de tensión y otras variaciones de voltaje, haciendo de esta familia de fuentes una herramienta de test ideal para ensayo de dispositivos en vuelo, pruebas en inversores y otros dispositivos que puedan experimentar interrupciones de voltaje. Las aplicaciones incluyen convertidores DC/DC, caída de tensión en inversores, simulación de arranque de motores, carga automática de baterías, ensayo de ciclos de vida de producto, etc.

La serie 62000P ofrece una amplia región de operación. Por ejemplo los rangos de trabajo del modelo 62012P-80-60 es de 0-80V y 0-60A con potencia máxima de 1200W, pudiendo trabajar en varias combinaciones según se muestra en la imagen. Las fuentes de alimentación convencionales proporcionan la misma corriente máxima en todos los valores de voltaje de salida, sin embargo, la serie 62000P pueden operar con corrientes mayores a voltajes más pequeños. Esto significa que un mismo dispositivo puede probarse con baja tensión/alta corriente y alta tensión/baja corriente usando una única fuente, ahorrando por lo tanto costes y espacio en el sistema de test automático.

Otra función incorporada en esta serie de fuentes CC es la programación secuenciada, con 100 secuencias programables por el usuario con configuración de tiempo desde 5ms a 15000s, control de relación de subida voltaje/corriente, salida TTL 8 bit y programación analógica con tarjeta interfaz de programación aislada para aplicaciones de test automático. Interfaces RS232 y USB estándar en toda la serie 62000P, y comunicación GPIB y/o Ethernet opcionales.

www.idm-instrumentos.es

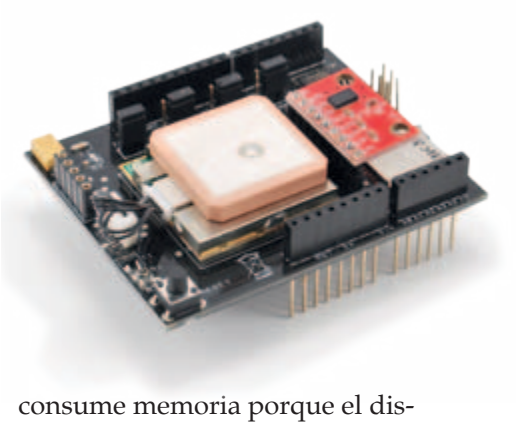
GPS LOGGER SHIELD

Mediante un simple sketch, registra periódicamente las posiciones obtenidas por un receptor GPS cuando está en movimiento y las guarda en una microSD formateada para que se pueda leer en un entorno Windows.

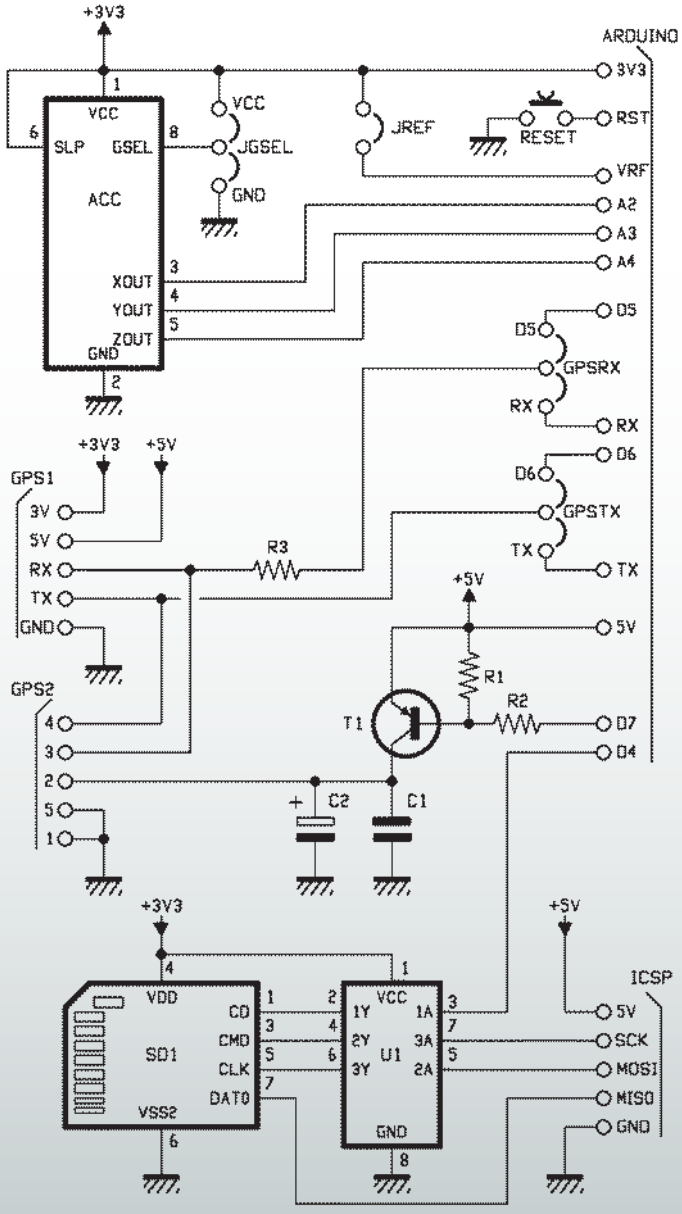
LABORATORIO INFORMATICO
FREE ED OPEN

El GPS (Global Positioning System) es el más conocido y utilizado de los sistemas de navegación por satélite y, también gracias a la integración con los sistemas de perfeccionamiento de la precisión (WAAS/EGNOS, DGPS, etc.) que se le han unido durante años, permite conocer en cada momento la posición y hora precisa (horario UTC) mediante el uso de módulos receptores fáciles de encontrar en el mercado, ahora a precios al alcance de todos. Muchos de estos receptores se comunican con el exterior enviando los datos obtenidos del sistema GPS a través del protocolo NMEA0183, que prevé una comunicación en serie y la emisión de paquetes de datos estándar fácilmente manejables por un microcontrolador, también para los relativamente limitados recursos de cálculo de Arduino. No es casualidad que justamente combinando un receptor GPS estándar NMEA con Arduino hayamos realizado el sistema propuesto en este artículo: se trata de un simple logger GPS, o lo que es lo mismo un dispositivo que permite trazar el recorrido realizado por una persona, un vehículo

o un bien, simplemente a través de la memorización periódica de puntos de localización obtenidos por el GPS mismo.
Para dejarnos ver este recorrido, el logger guarda el trazado en forma de lista de registros (que contienen los datos de las posiciones registradas) en una tarjeta de memoria microSD, desde la cual se puede pasar a un ordenador personal para tener la traza de sus propios viajes, pero también quizás, para contribuir a proyectos open source de seguimiento de mapas alternativos a Google Maps, como por ejemplo Openstreetmap (*www.openstreetmap.org*).
Teniendo una discreta experiencia con los logger GPS disponibles en el mercado, nos hemos dado cuenta que una de las cosas más molestas es olvidarse de apagarlos una vez llegados a nuestro destino; de hecho esto



consume memoria porque el dispositivo guarda continuamente la misma posición, o mejor dicho, muchos puntos muy cercanos entre sí, cuya distancia no es debida al movimiento del receptor sino al error de localización por el cual el receptor está afectado y que no se repite en más lecturas de las mismas coordenadas. El error en el posicionamiento depende de factores atmosféricos, de retrasos de propagación típicos de la lógica y de la tolerancia del reloj local (que no es tan preciso como el atómico de los satélites GPS); la variación de los factores en juego hace que aún manteniendo parado el receptor, este proporcione posiciones que varían algunos metros una de otra, dando la indicación que el receptor se mueve aunque sin embargo está parado.
En todo caso, para evitar enviar inútilmente datos sobre la localización a la microSD aún con el receptor parado, hemos decidido implementar en nuestro logger un firmware capaz de parar el registro, o reanudar cuando el sistema se pone de nuevo en movimiento; esto se ha logrado gracias a un acelerómetro, cuyas informaciones elaboradas por Arduino, además de permitirnos interrumpir el salvado de los datos cuando el circuito está parado, nos permite apagar el módulo receptor GPS para reducir el consumo, cosa muy útil, dado que por norma este tipo de sistemas se alimenta por batería.



[esquema **ELÉCTRICO**]

EL HARDWARE

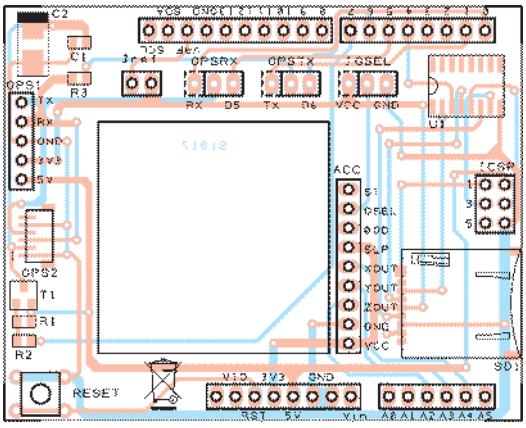
El logger que os proponemos consta de un shield, sobre el cual se encuentra principalmente el receptor GPS, el lector de SD-Card y el acelerómetro, además de una tarjeta Arduino UNO; hay que señalar que el receptor se conecta mediante dos conectores (alternativos el uno del otro), con lo que nada impide situarlo fuera del shield. Esto puede servir si sobre el circuito pensáis montar otros shields, que inevitablemente perturbarían la recepción de las señales provenientes de los satélites.

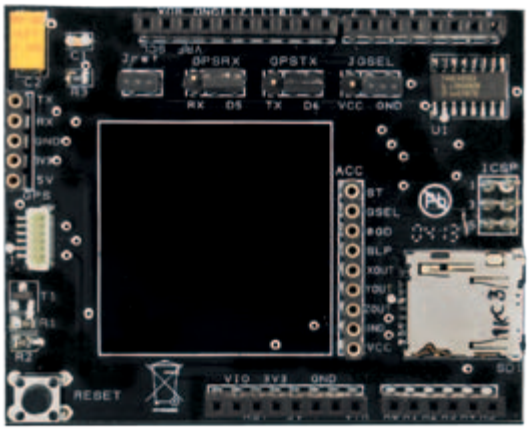
Veamos entonces en qué consiste el circuito del shield, que gracias a la reducida absorción de corriente, se alimenta directamente desde el Arduino. En nuestras mediciones, alimentando Arduino a 5 V, indican consumos de alrededor a 35 mA con el GPS activo, reducido a cerca de 5 mA desactivando el GPS a través del pin 7. A esto se suma los cerca 40 mA del Arduino UNO, con lo que tenemos un total de 75 mA en la modalidad LOG (durante la adquisición y registro de las coordenadas) y 45 mA en modo standby.

La sección principal del shield está compuesta obviamente por el módulo GPS EM406 de la GlobalSat, dotado de antena integrada (la tiene encima...) que comunica con Arduino a través del protocolo NMEA 0183: la comunicación es serie a 4.800 bps mediante líneas TX e RX a nivel TTL. Esta señal en serie está disponible sobre el conector GPS y a través de los jumper GPSRX y GPSTX puede ser direccionada a los pin 0 y 1 del Arduino (RX y TX del puerto serie integrada) o también 5 y 6; respecto a esto recordamos que las líneas 0 y 1 constituyen el UART físico de Arduino, aunque no esté disponible porque está siendo usado

La típica "nube" de puntos cercanos que se obtiene con pocos minutos de diferencia. En espacios cerrados, el área cubierta es aún más amplia porque aumenta el error.

[plano de **MONTAJE**]





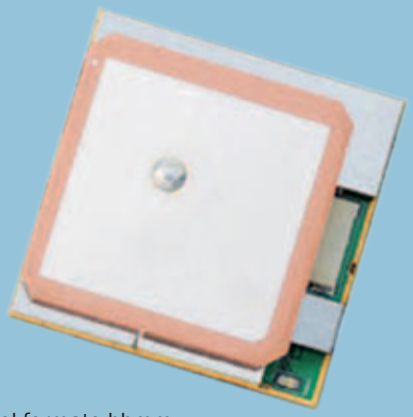
Lista de materiales

R1: 10 kohm (0805)
R2: 4,7 kohm (0805)
R3: 2,2 kohm (0805)
C1: 100 nF multicapa (0805)
C2: 470 µF 6,3 VL tántalo (D)
T1: BC807
U1: 74HC4050D

GPS: Receptor EM406
SD1: Conector micro SD-Card
RESET: Micro interruptor
ACC: MMA7361acelerómetro 3 ejes

Varios:
- Conector 6 polos paso 1mm (cod. BM06B-SRSS-TB/CONNEM406A)
- Tira de 3 pines Macho/Hembra (2 pz.)
- Tira de 6 pines Macho/Hembra (1 pz.)
- Tira de 8 pines Macho/Hembra (2 pz.)
- Tira de 10 pines Macho/Hembra (1 pz.)
- Tira de 2 pines Macho 2 pines (1 pz.)
- Tira de 3 pines Macho 3 pines (3 pz.)
- Tira de 5 pines hembra 5 pines (1 pz.)
- Tira de 9 pines hembra 9 pines (1 pz.)
- Jumper (4 pz.)
- Circuito impreso

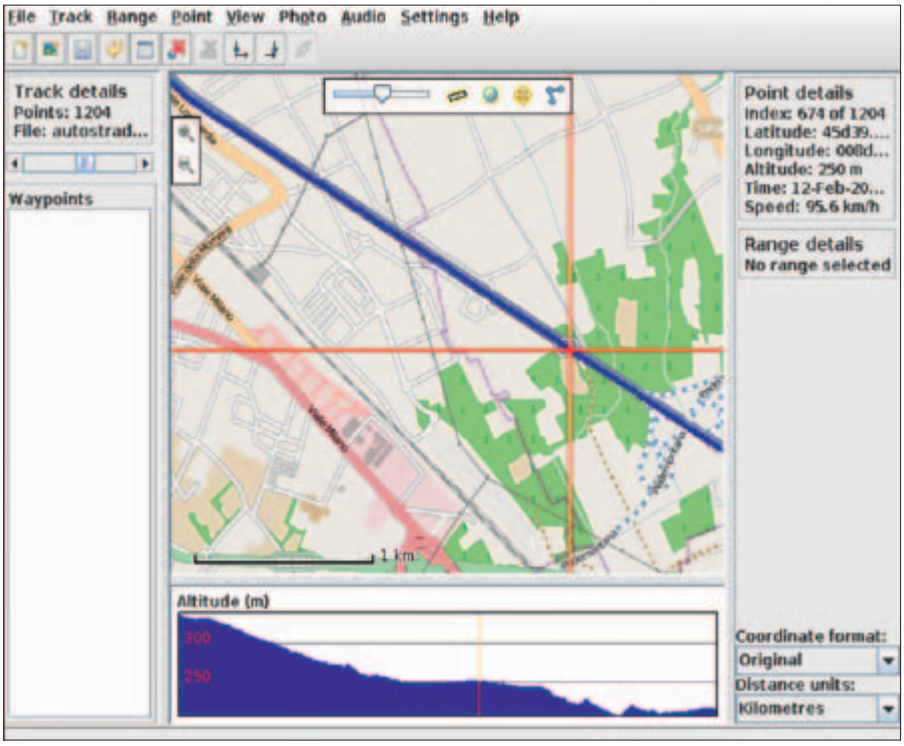
El estándar **NMEA0183**



El protocolo NMEA ha sido desarrollado para las comunicaciones entre dispositivos electrónicos utilizados en ámbito naval; sus especificaciones están disponibles con coste en la web de la National Marine Electronics Association, sin embargo el protocolo ha sido reconstruido por varios aficionados usando fondos públicos, por lo que se encuentra libre en la web, por ejemplo en las direcciones https://en.wikipedia.org/wiki/NMEA_0183 y www.gpsinformation.org/dale/nmea.htm.
El protocolo NMEA prevé una comunicación serie a 4.800 bps, 8 bit de datos sin bit de paridad y con un bit de stop.
Cada paquete de datos transmite "frases" que empiezan con el carácter **$**, terminan con la secuencia **<CR><LF>** (también escrita \r\n) y contienen campos separados por comas. El último campo, opcional, está separado por un * y contiene dos cifras hexadecimales de checksum de los caracteres de la frase comprendidos entre $ y *, extremos excluidos.
El primer campo de la frase identifica el tipo; generalmente inicia con dos caracteres que identifican el dispositivo (GP para un receptor GPS), o con P para las frases propietarias definitivas de los productores de varios dispositivos. El módulo GPS de nuestro shield envía principalmente las tres frases más importantes: GGA (Global Positioning System Fix Data, los datos de posición y exactitud), RMC (Recommended Minumum, los datos de posición más importantes) y GSA (datos sobre los satélites).
Los campos de la frase GGA son:

- GPGGA (Global Positioning System Fix Data);
- hora (UTC) del fix, en el formato hhmmss;
- latitud ddmm.mmm por dd° mm.mmm';
- N o S;
- longitud ddmm.mmm por dd° mm.mmm';
- E o W;
- calidad del fix, cuyos valores más importantes son 0 por fix no válido y 1 por fix GPS;
- número de satélites;
- "HDOP", un medida del error;
- altura sobre el nivel medio del mar;
- altura del geoide respecto al elipsoide WGS84;
- tiempo en segundos desde la última actualización DGPS;
- ID de la estación DGPS;

ya por otras aplicaciones (por ejemplo porque habéis montado otros shield) podéis, mediante la librería SoftwareSerial, configurar vuestra tarjeta para que gestione las líneas 5 y 6 como un UART virtual.
El módulo GPS se activa ajustando el contacto 7 a nivel lógico bajo: desde ese transmitirá cíclicamente paquetes de datos NMEA en serie hasta cuándo será desactivado ajustando el mismo contacto 7 a nivel alto.
En lo que se refiere al slot microSD, el shield incluye el convertidor de niveles 74HC4050D para permitir la comunicación vía SPI (de hecho las SD-Card y las microSD utilizan para la comunicación con el exterior el bus Serial Peripheral Interface) a niveles 0/3,3V con la tarjeta de memoria. Las señales correspondientes se conectan al conector que en Arduino UNO corresponde al ICSP, de manera que permite la compatibilidad también con las tarjetas Arduino MEGA. Entonces los contactos laterales del shield relativos a las líneas 10, 11, 12, 13 de Arduino UNO (que repiten el conector ICSP) se conectan con el resto del circuito; hay que tener presente que sí a Arduino se el conectan en cascada otros shield, no podéis seguir utilizando las mencionadas 10, 11, 12, 13, porque están



Mapa de referencia proveniente de OpenStreetMap, licencia CC-BY-SA. http://www.openstreetmap.org/copyright.

• checksum.
Los campos de la frase RMC son:
• RMC (Recommended Minimum sentence C);
• hora (UTC) del fix, en el formato hhmmss;
• status del fix: A para “Active” (valido) V para “void” (invalido);
• latitud ddmm.mmm para dd° mm.mmm’;
• N o S;
• longitud ddmm.mmm para dd° mm.mmm’;
• E o W;
• velocidad respecto al suelo en nudos;
• dirección de viaje en grados;
• fecha en formato ddmmyy;
• variación magnética;
• tipo de fix: A para autónomo, D para diferencial o otros valores para fix no fiables;
• checksum.
Los campos de la frase GSA son:
• GPGGA;
• selección del fix 3D o 2D (A para auto, M para manual);
• tipo del fix: 1, 2 o 3 para ningún fix, fix 2D y fix 3D respectivamente;
• 12 campos para los PRN de los satélites usados para el fix;
• PDOP (una medida del error);
• HDOP (una medida del error horizontal);
• VDOP (una medida del error vertical);
• checksum.
Como ejemplo, a continuación mostramos el código impreso al encender nuestro módulo, con frases del chipset seguidas por frases vacías antes de encontrar el fix:

```
$PSRFTXT,Version:GSW3.5.0_3.5.00.00-SDK-3EP2.01 *46
$PSRFTXT,Version2:F-GPS-03-1006232*29
$PSRFTXT,WAAS Disable*13
$PSRFTXT,TOW:  237406*11
$PSRFTXT,WK:   1727*66
$PSRFTXT,POS:  6378137 0 0*2A
$PSRFTXT,CLK:  96250*25
$PSRFTXT,CHNL: 12*73
$PSRFTXT,Baud rate: 4800*65
$GPGGA,175631.867,,,,,0,00,,,M,0.0,M,,0000*58
$GPGSA,A,1,,,,,,,,,,,,,,,*1E
$GPRMC,175631.867,V,,,,,,,120213,,,N*40
```

Abajo, un extracto de datos NMEA una vez conocida la posición:

```
$GPGGA,175802.000,4546.1373,N,00848.7133,E,1,06,1.7,335.
7,M,48.0,M,,0000*5C
$GPGSA,A,3,15,09,05,08,26,28,,,,,,,2.8,1.7,2.3*31
$GPRMC,175802.000,A,4546.1373,N,00848.7133,E,37.18,190.
20,120213,,,A*54
```

ya utilizadas. Estas líneas pueden ser utilizadas como GPIO, o para gestionar una comunicación sobre bus SPI, no ambas cosas juntas.
De Arduino se puede acceder a la microSD, formateada con file-system FAT16 o FAT32, usando la librería SD presente en el entorno Arduino, teniendo cuidado de usar *SD.begin(4)* para configurar el Chip Select distinto del estándar.
Y pasamos al acelerómetro, con las siglas ACC en el esquema eléctrico: es del tipo de tres ejes y se trata del MMA7361 producido por Freescale; de este componente usamos la versión producida por Sparkfun, ya montada sobre un soporte que lleva 9 patas de conexión con una separación de 2,54mm. Este módulo es alimentado a 3,3 V por Arduino a través de la línea 3.3 V y GND, que sobre el MMA7361 Sparkfun alcanzan a los conectores VCC y GND respectivamente. El acelerómetro proporciona en su salida una triple señal analógica de la aceleración de cada uno de los ejes; las señales están disponibles en el pin 4 para el eje X, el3 para el eje Y y el 2 para el eje Z.
A través del jumper Jref es posible conectar el VREF del Arduino a los 3,3 V, de manera que el valor de 1,65 V en salida del acelerómetro, correspondiente a aceleración nula (0g), coincida más o menos con el valor 512 a la salida del convertidor A/D a 10 bit de Arduino, es decir, la mitad del rango del convertidor (el A/D que varía entre 0 y 1024)
El jumper JGSEL permite seleccionar una de las dos modalidades de funcionamiento del acelerómetro: conectado a GND tendrá escala completa de 1.5 g y resolución de 800 mV/g, correspondiente a un valor AnalogRead de alrededor 248 por g; conectado a VCC el fondo de escala pasa a 6 g y la resolución 206 mV/g, correspondiente a una lectura de alrededor 64 por g. La configuración predefinida por nuestro logger prevé que los jumper centrales del shield, GPSRX y GPSTX, estén siempre cerrados sobre los pines de la derecha (respectivamente D5 y D6) mientras el jumper JGSEL, relativo al acelerómetro, este cerrado a GND; además el jumper Jref debe estar cerrado.

EL FIRMWARE

Por sencillez, hemos decidido guardar en la memoria los paquetes de datos NMEA recibidos del módulo GPS, también para ahorrar espacio respecto a los formatos más utilizados como GPX y KML, que en todo caso se pueden obtener a partir de nuestros datos, mediante programas especiales.
Para gestionar la comunicación con el módulo GPS usaremos SoftwareSerial; si la tarjeta de memoria microSD no está presente en el circuito , los datos del GPS serán reportados sobre la salida serie del Arduino, de manera que se puedan pasar directamente a un ordenador, tanto para la depuración como para eventuales aplicaciones avanzadas. El control del movimiento a través acelerómetro se produce aproximadamente cada

Listado 1

```
#include <SD.h>

#include <SoftwareSerial.h>

/* ****** Settings ****** */
char log_filename[13] = "data.log"; // keep this FAT friendly
// how long should we wait when not moving before we stop logging
#define STOP 60000
/* ****** End of user settings ****** */

#define PIN_GPS_ENABLE 7
SoftwareSerial gpsSerial(5,6);

#define PIN_SD_SS 10
File log_file;
boolean sd_available = false;

boolean moving = true;
unsigned long last_move = 0;
#define ZERO_X 512
#define ZERO_Y 512
#define ZERO_Z 512
#define M_THRESH 60000
#define PIN_X 4
#define PIN_Y 3
#define PIN_Z 2

void setup() {
    Serial.begin(9600);

    //GPS setup
    gpsSerial.begin(4800);
    pinMode(PIN_GPS_ENABLE,OUTPUT);
    digitalWrite(PIN_GPS_ENABLE,HIGH);

    //SD setup
    pinMode(PIN_SD_SS,OUTPUT);
    pinMode(10,OUTPUT);
    start_sd();

    start_gps();
}

void loop() {
    if(moving) {
        if(sd_available) {
            log_file = SD.open(log_filename,FILE_WRITE);
            if(!log_file) {
                sd_available = false;
            }
        }
        if(gpsSerial.overflow()) {
            if(sd_available) {
                log_file.write("\r\n$PXXXX,SoftwareSerial overflow!!!");
            } else {
                Serial.println("\r\n$PXXXX,SoftwareSerial overflow!!!");
            }
        }
        while(gpsSerial.available()) {
            int r = gpsSerial.read();
            if(sd_available) {
                log_file.write(r);
            } else {
                Serial.write(r);
            }
        }
        check_movement();
        log_file.close();
    } else {
        delay(1000);
        check_movement();
    }
}

void start_gps() {
    gpsSerial.listen();
    digitalWrite(PIN_GPS_ENABLE,LOW);
}

void stop_gps() {
    digitalWrite(PIN_GPS_ENABLE,HIGH);
}
```

(Continúa)

segundo; en el caso que se perciba un movimiento (en cualquier dirección), el logger queda activo o es inmediatamente activado, mientras los movimientos estén por debajo de cierto umbral (M_THRESH) para todos los controles efectuados durante un minuto (configurable a través del valor STOP, en milisegundos) el GPS está apagado.

Los valores presentes en el sketch han sido controlados empíricamente para proporcionar resultados aceptables para un uso típico.

Por comodidad, en fase de depuración, cuando dejamos de guardar los datos, lo escribimos en el archivo NMEA, usando una frase identificada por el string PXXXX, que será debidamente ignorada por los programas. El mismo string se usará para señalar eventuales desbordamientos de la SoftwareSerial, que podrían ser fuente de problemas.

El archivo sobre SD (en nuestro ejemplo DATA.LOG) es cerrado frecuentemente, de manera que en cualquier momento sea posible quitar la alimentación, mover la microSD en un lector conectado a un ordenador y obtener el registro del recorrido realizado.

El firmware para cargar en Arduino para hacer funcionar el logger GPS se encuentra en el **Listado 1**.

FUTUROS DESARROLLOS

Una personalización simple del logger consiste en integrar algunos sensores ambientales para añadir informaciones al nuestro log de viaje: por ejemplo para relacionar con la posición ciertas condiciones de humedad, temperatura, viento, para así obtener una geolocalización de los datos.

Más adelante explicaremos como interpretar las frases NMEA de Arduino, de manera que po-

damos realizar proyectos más complejos que un simple almacenamiento de los datos.

GESTIONAR EL ARCHIVO NMEA

Los archivos NMEA guardados por nuestro logger se pueden leer directamente solo por un número limitado de programas. Per suerte existe GPSBabel, un programa libre y multiplataforma que permite convertir desde y hacia la mayor parte de los formatos usados por sistemas GPS. GPSBabel puede ser descargado desde el sitio oficial, *https://www.gpsbabel.org/* para Windows y OS X (los usuarios Linux pueden encontrarlo en el repositorio de su distribución): usarlo es simple, basta seleccionar el formato de entrada (en nuestro caso NMEA), el nombre del archivo a leer (DATA.LOG, en la microSD), el formato a generar (por ejemplo GPX para la mayor parte de los programas o KML para Google Earth) y el nombre del archivo a escribir. GPSBabel ofrece innumerables opciones para definir la conversión, pero la configuración predefinida está generalmente adaptada a la mayor parte de los usos.
Otro programa que puede ser útil es GpsPrune, también libre y multiplataforma (escrito en Java), que permite visualizar, convertir y manipular trazas GPS usando come escenario los mapas libres del proyecto OpenStreetMap, y que soporta directamente el formato NMEA. GpsPrune se puede descargar desde la web oficial *http://activityworkshop.net/software/gpsprune/* y está también disponible en los repositorios de muchas distribuciones Linux. Para abrir nuestro archivo será necesario renombrarlo con extensión **.nmea**, de manera que permita el reconocimiento automático del formato, entonces podremos abrirlo desde el menú *File->Open File*.

Listado 1 – continuación

```
void start_sd() {
    if (!SD.begin(PIN_SD_SS)) {
        Serial.println("Card failed, or not present");
    } else {
        log_file = SD.open(log_filename,FILE_WRITE);
        if (log_file) {
            log_file.close();
            sd_available = true;
        } else {
            Serial.print("Can't open file <");
            Serial.print(log_filename);
            Serial.println("> for writing.");
        }
    }
}

void check_movement() {
    int x = analogRead(PIN_X) - ZERO_X;
    int y = analogRead(PIN_Y) - ZERO_Y;
    int z = analogRead(PIN_Z) - ZERO_Z;
    long acc = (long)x*x + (long)y*y + (long)z*z;
    if (acc > M_THRESH) {
        last_move = millis();
        if (! moving) {
            start_gps();
            start_sd();
            // Ignore data that has overflowed
            gpsSerial.overflow();
        }
        moving = true;
    } else {
        if (moving) {
            unsigned long now = millis();
            // TODO: check for the overflow condition of millis
            // (or reboot the arduino every 49 days :) )
            if (now-last_move > STOP) {
                moving = false;
                stop_gps();
                if(sd_available) {
                  log_file.write("\r\n$PXXXX,Device is idle, stop logging\r\n");
                } else {
                 Serial.println("\r\n$PXXXX,Device is idle, stop logging\r\n");
                }
            }
        }
    }
}
```

REALIZACIÓN PRÁCTICA

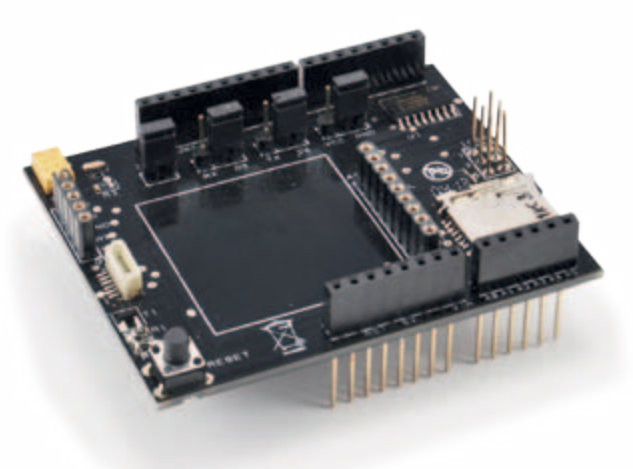
La construcción del shield es bastante simple: una vez obtenida la placa (los archivos de diseño están disponibles en nuestra pá-

Las líneas de **Arduino**

Estas son las líneas de Arduino que interesan al logger GPS.

0 = RX Serie
1 = TX Serie
4 = SD Chip Select
5 = SoftwareSerial RX
6 = SoftwareSerial TX
7 = GPS enable
10 = SPI SS (non usato)
11 = SPI MOSI 1
12 = SPI MISO 1
13 = SPI SCK 1
A2 = Acc Z
A3 = Acc Y
A4 = Acc X

Sobre el shield, las 10, 11, 12, 13 no están conectadas porque para la comunicación SPI usamos las correspondientes del conector ICSP; esta disposición nos permite usar el shield también sobre Arduino MEGA, que tiene el conector ICSP en la misma posición y cuyas líneas SPI se repiten sobre las 50, 51, 52, 53. Como las líneas ICSP están en paralelo con ellas, no es posible emplearlas como GPIO; sobre Arduino MEGA es sin embargo posible utilizarlas, pero solo para comunicaciones con otros dispositivos SPI.



gina web) montaros primero el lector de microSD, en el que hay que aplicar el estaño a las pines y las lengüetas de fijación que servirán a mantenerlo bien sujeto durante la inserción y extracción de la tarjeta de memoria. Soldar después el integrado 74HC4050D, posicionándolo centrado con los correspondientes pads de soldadura, fijándolo con una gota de estaño un par de pines en los esquinas y después procediendo con los pines restantes.
Para las conexiones con el receptor GPS, dependiendo del modelo que tengas, el conector hembra SIL de paso 2,54 mm o el conector miniatura 6 polos SIL de paso 1,25 mm.
Para el módulo acelerómetro usar una tira de 9 contactos hembra de paso 2,54 mm, mientras para los jumper emplearemos tiras de pines macho de paso 2,54 mm.
Los conectores para la inserción en el Arduino son las habituales tiras hembra (una de 6, dos de 8 y una de 10 contactos, al menos si queréis la compatibilidad con Arduino UNO) con pines de largo 20 mm. Para la conexión sobre ICSP sirve además un conector 6x2 polos de paso 2,54 mm con pin de largo 20 mm.

(180023)

el MATERIAL

El shield GPS ya montado y probado está disponible al precio de 22,00 Euros (FT1017M); el shield no incluye el receptor GPS con antena integrada (cod. EM406A, Euro 45,00), ni el acelerómetro a tres ejes (cod. MMA7361, Euros 16,00).

Precios IVA incluido sin gastos de envío.
Puede hacer su pedido en:
www.nuevaelectronica.com
pedidos@nuevaelectronica.com

FRECUENCIMETRO DIGITAL
BASADO EN MICROCONTROLADOR

En esta segunda y última entrega, describiremos el montaje en la caja, el firmware y la prueba final de nuestro instrumento de medida de la frecuencia de señales analógicas de BF hasta 10 MHz, TTL y CMOS hasta 50 MHz, y de radiofrecuencia hasta 1,1 GHz.

MICHELE MENNITI

Ya hemos explicado el esquema eléctrico y el montaje de los tres circuitos impresos que constituyen las tres unidades de nuestro frecuencímetro (Sección de alimentación, Sección lógica y display LCD, Sección entradas). También hemos estudiado el comportamiento de varias señales aplicadas a las tres entradas, mediante imágenes tomadas del DSO, de tal manera que hemos podido evaluar las notables cualidades de nuestro frecuencímetro digital.
En esta última parte veremos cómo montar todo el frecuencímetro en una caja idónea, de qué manera programar el microcontrolador que controla el instrumento y como utilizarlo una vez terminado.
La caja que hemos previsto para alojar el frecuencímetro es el modelo 5100-AUS12 de Teko: está realizado con las tapas de plástico, los paneles anterior y posterior en aluminio y dispone de ranuras de ventilación en las cubiertas superiores e inferiores; dispone además de distintos puntos de sujeción en el fondo, útiles para la fijación de los circuitos impresos.
Antes de hablar de la caja es necesario retomar el tema de la entrada de alimentación de 220 Vac, que dejamos pendiente: como ya hemos dicho, la radiofrecuencia tiende a infiltrarse por todas partes, creando trastornos de todo tipo al instrumento y también a otros aparatos localizados en el mismo local. Esto es porque muchas veces el RF consigue llegar incluso hasta la red eléctrica y, a

Fig. 1 *- El panel posterior completo.*

través del sistema, todos los aparatos a ella conectada. Hemos trabajado mucho para bloquear cada intento de la RF por infiltrarse en los circuitos que componen el frecuencímetro y no podíamos descuidar la protección de la red eléctrica. La solución que os proponemos es extremadamente simple: se trata de desmontar de una vieja alimentación de un ordenador la toma de red con filtro de la que dispone y todo lo que está conectado a ella (bobinas, condensadores, resistencias, anillos de ferrita) incluidos los hilos que van sobre el circuito impreso, que en general son de buena calidad y sección adecuada. Para el acoplamiento de la toma sobre el panel posterior necesitaremos trabajar con mucha paciencia, pero una vez montada dará al instrumento el aspecto profesional que merece.

Los dos hilos provenientes de tal toma se conectarán a los extremos de un interruptor bipolar, idóneo para conmutar 220 Vac que puede tomarse de la misma alimentación de PC, siempre que sea de tipo bipolar, de manera que interrumpa ambos cables de la corriente eléctrica. De hecho, en nuestras instalaciones no hay una referencia fija para la fase y el neutro, y como aquella y como la que debe desconectarse es siempre la fase, ante la duda es preferible desconectar ambos hilos. El interruptor también se fijará en el panel posterior de la caja. Los otros dos extremos del interruptor bipolar obviamente se conectaran al conector J1 del PCB del punto de alimentación. Las tomas 220V son siempre de tres polos, el tercero de los cuales es la tierra y a ella, en el 99% de los casos, se conectan un cable amarillo/verde que termina con un ojal.

Si, como es deseable y requiere la ley vigente, vuestro sistema eléctrico está dotado de toma a tierra, debéis conectar el ojal de la toma al panel de aluminio posterior, fijándolo con un tornillo más adecuado. En caso contrario, cubrirlo completamente con cinta aislante de buena calidad y dejarlo libre.

Si no disponéis de una alimentación de PC para desmontar, usar un filtro de red de pequeña escala (bastan 0,5 amperios), por ejemplo aquellos productos de Schaffner (*www.schaffner.com*). Los filtros de red tienen dos contactos de entrada (más dos láminas de masa, que van al metal), a las que hay que soldar una toma tripolar de panel, o directamente el cordón de alimentación, y dos de salida, que conectareis al interruptor bipolar. Los hay también dotados de toma tripolar incorporada, que seguramente son preferibles.

De todas maneras, las conexiones de la sección de entrada 220Vac se hacen como se muestra en la **Fig. 3** de la primera parte (también la **Fig. 3** de este número debería ayudar) donde veréis que todos los hilos que llevan corriente alterna han sido perfectamente aislados con manguito termorretráctil, operación que os aconsejamos encarecidamente realizar, con el fin de evitar la descarga eléctrica mientras manejáis el frecuencímetro con la tapa abierta.

En la **Fig. 1** podéis ver el panel posterior con el tornillo de la puesta a tierra, la toma y el interruptor.

El trabajo más delicado y laborioso es sin duda la preparación del panel anterior, porque albergara el display LCD, los dos pulsadores de selección entrada y rango de medida, las tres tomas BNC, la toma SMA, los tres LED que indican la activación de la entrada.

Es necesario primero colocar físicamente los PCB Logica y Display y Entradas en la caja y después empezar con los bordes del display, la posición precisa de los dos pulsadores y de la toma SMA (soldada directamente sobre el PCB); conviene marcar estos puntos en la parte interna del frontal y empezar con agujeros muy pequeños (aconsejamos una broca de

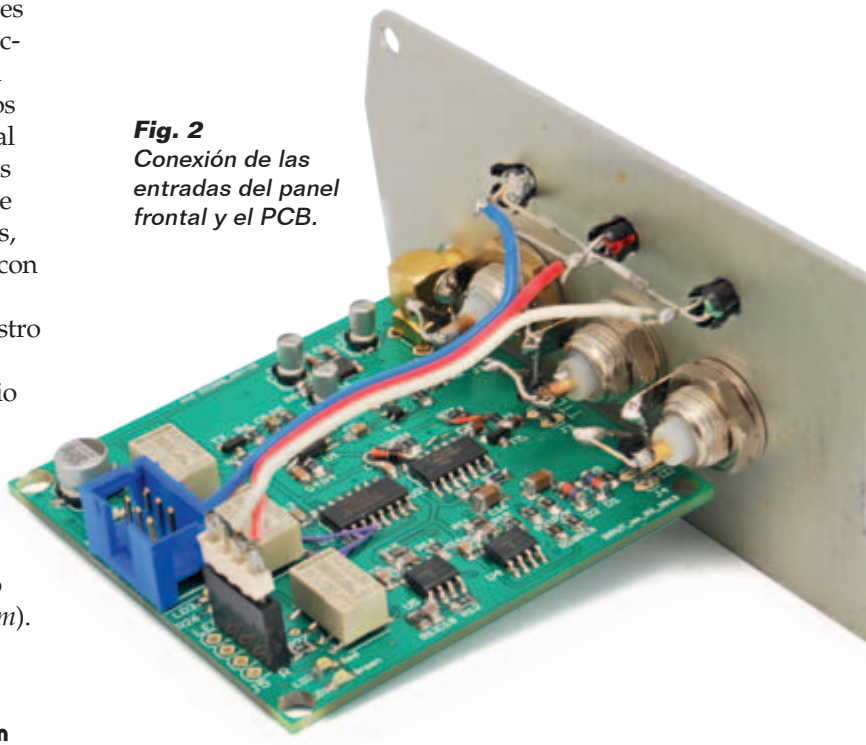


Fig. 2
Conexión de las entradas del panel frontal y el PCB.

1 mm), de manera que verifiquemos la precisión de las posiciones. Solo después se podrán usar brocas cada vez mayores, así conseguiremos hacer agujeros precisos. También para el display, conviene realizar un rectángulo más pequeño respecto a las dimensiones reales del display, así será más fácil trabajarlo con una lima plana de manera que realicemos contornos precisos.
Más simple será la perforación relativa a las tres tomas BNC y a los tres LED, ya a que estos seis elementos estarán conectados al PCB de la Sección de entradas mediante trozos de hilo rígido; la única cosa a tener en cuenta, será la equidistancia entre los tres BNC y entre los tres LED. Además será necesario poner atención en la perfecta alineación entre cada LED y su respectivo BNC.
Los conectores BNC deben estar dotados del ojal para la conexión de las masas; el polo central de cada uno estará conectado a la plataforma de su respectiva entrada mediante un trozo de hilo rígido, mientras el ojal estará conectado al plano de masa (GND). Para simplificar esta operación, y también otros sucesivos desmontajes del PCB de las entradas, nosotros hemos soldado sobre las tres parejas de la placa otras tantas parejas de pines macho, como se muestra en la **Fig. 2**.
Respecto a los tres LED de 3 mm, sus cátodos estarán conectados juntos y después al ojal de la toma BNC "TTL", y a cualquier punto GND, mientras los tres ánodos se han conectado a su respectivo conector mediante un pequeño conector macho. El conector dispone, como se ha dicho, también la conexión para el LED amarillo, que en nuestro prototipo no hemos llevado este LED al panel, nos hemos contentado con dejarlo sobre el PCB para las verificaciones funcionales necesarias. El conector está preparado para llevar dos hilos por cada LED, así que si no queréis adoptar nuestra solución de conexionado, podéis usar una pequeña base de seis hilos (ocho, si queréis llevar también el LED amarillo) para conectar ánodos y cátodos directamente al PCB; para la conexión seguir atentamente la imagen del plano de montaje del PCB y la serigrafía: el ánodo de cada LED está indicado con la inicial del color (B=blue, Y=yellow, R=red, G=green).
Sea como sea, conviene emplear un conector header hembra en el PCB y una tira de pines macho para la conexión; en el caso que debáis desmontar el PCB, en vez de desoldar todo bastará con el quitar la tira de pines. Parar fijar de manera estable los LED de 3 mm al panel anterior, aconsejamos utilizar los porta-led de metal o plástico; en caso contrario, probablemente debáis recurrir a un hilo de pegamento de fusión o silicona. Observando la **Fig. 3** podréis obtener ideas

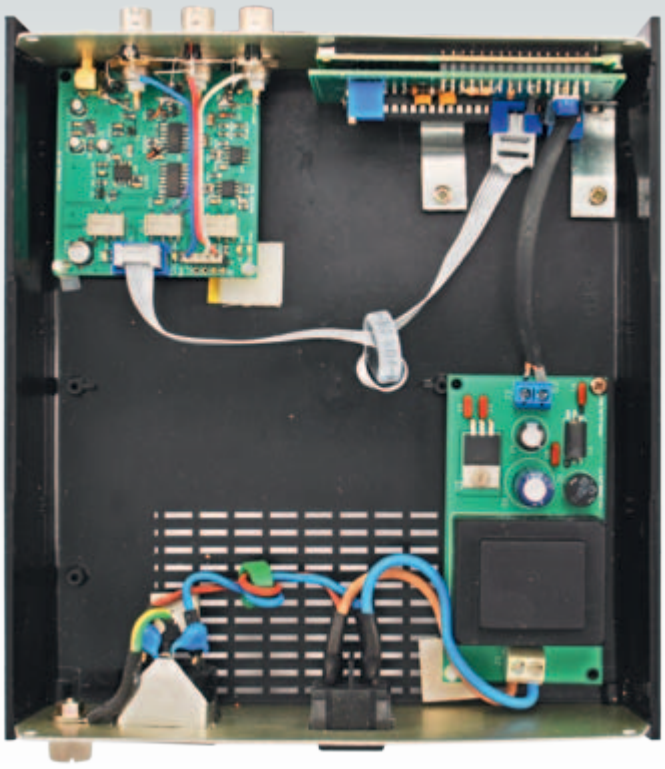


Fig. 3 *- Fijación de los tres PCB en la caja.*

para la fijación de los tres PCB; nosotros hemos usado tornillos, distanciadores de plástico (con o sin base adhesiva) y conectores de aluminio; en conclusión, lo importante es colocar correctamente los PCB y hacer de manera que no se muevan ni siquiera girando la caja boca bajo o sacudiéndola.

EL FIRMWARE

La realización de este proyecto ha sido posible fundamentalmente gracias a la librería "Frequency Counter", de la firma de Martin Nawrath de la Academy of Media Arts–Cologne, la cual se ocupa de toda la gestión necesaria de los temporizadores *(timer)* del micro, con la finalidad de calcular la frecuencia aplicada al pin 11. La librería es descargable del link original *http://interface.khm.de/index.php/lab/experiments/arduino-frequency-counter-library/* y se guarda en su propia

Como gestiona los pulsadores el **watchdog**

Cada vez que se presiona P1, el firmware verifica si este ha sido pulsado anteriormente y, en caso afirmativo, cuanto ha durado tal presión y cuanto hace que ha sucedido; de esto se encarga el debounce software basado en el cálculo del número de los desbordamientos (overflow) del contador del watchdog. En el caso en el cual no hayan pasado el tiempo de debounce previsto, el firmware reconoce la presión como un rebote del contacto y la ignora. Sin embargo si ha pasado al menos dicho tiempo, entra en la rutina de gestión del pulsador, ósea la rutina de interruptor "entrada" que incrementa la variable SW_COUNT (de 1 a 3, después se restea); en base al valor de esta última, viene seleccionada la correspondiente entrada (BF, TTL, RF), mediante la oportuna combinación de los tres relé de la Sección entrada. Al tiempo se encenderá el LED que indica cual de los tres BNC del panel esta activo. Del mismo modo, cada vez que se pulsa P2 se activa la rutina de interruptor "rango" que incrementa la variable HZ_COUNT (de 1 a 4, después se restea); en base al valor de esta última, se selecciona el rango correspondiente (Hz, kHz, MHz, GHz) que da origen a una conversión software de la frecuencia medida en base al rango elegido. Por ejemplo, si se mide una frecuencia de 1 MHz, cambiando las diferentes rangos se obtendrán las siguientes visualizaciones: 1000000 Hz, 1000.000 kHz, 1.000000 MHz, 0.001000000 GHz.
Y veamos ahora que sucede una vez que se ajusten la entrada y el rango deseados. Como el programa trabaja en un bucle infinito y efectúa una lectura cada segundo de la entrada del micro (pin 11) y proporciona los datos en base a los ajustes en la rango, si durante un ciclo se presiona P1 y/o P2, la lectura se falsea; por lo que es necesario siempre esperar un par de lecturas nuevas antes de poder estar seguros de lo que se lee en el display. Lo mismo es aplicable cuando se cambia el punto de medida o el valor a leer: de hecho es prácticamente imposible que la llegada de la nueva frecuencia coincida con el inicio de un ciclo de lectura.
Observando las pocas líneas presentes en el bucle (loop) del código se puede ver que, una vez realizada la lectura de la frecuencia, se siguen estas rutinas:

- Checkingr lee el estado de los tres relés y determina la proveniencia de la señal de entrada;
- Calcport convierte el valor numérico de la frecuencia en un string, de manera que simplifica los cálculos en base al rango de medida seleccionado;
- Backlight gestiona la retroiluminacion, manteniéndola a baja luminosidad en ausencia de señal de entrada y llevándola a alta luminosidad cuando es detectada una frecuencia cualquiera;
- Lcdprint prepara el string definitivo y lo visualiza sobre el display LCD, mostrando a la vez el rango y la entrada selec-

carpeta personal o en aquella original de las librerías de Arduino, e incluirla en el sketch junto a las otras librerías necesarias ("LiquidCrystal" para la gestión del LCD, "avr/wdt" para la gestión del watchdog, ambas incluidas en el IDE versión 1.0.1 o sucesivas). Observando el firmware (descargable desde la página de la revista 320 en www.nuevaelectronica.com), después de la inclusión de las librerías encontrareis el listado de las constantes y variables usadas en el sketch; como de costumbre, el sketch está muy bien comentado, por lo que no tendréis ninguna dificultad para seguirlo y comprenderlo: por esta razón no lo explicaremos paso-paso, y solo por secciones lógicas. El firmware se encarga de gestionar las funcionalidades de los dos pulsadores P1 e P2 que sirven, respectivamente, para elegir la entrada activa entre BF, TTL y RF y para cambiar el rango de medida, entre Hz, kHz, MHz y GHz. Debido a que los timer del micro son utilizados por la librería del frecuencímetro, se ha recurrido al uso del watchdog (véase el recuadro correspondiente) para gestionar correctamente el debounce (anti rebote) y garantizar el perfecto funcionamiento de los dos pulsadores. El rebote es el fenómeno por el cual, a la simple y única presión de un pulsador, en realidad se producen varios contactos; cuando se trabaja en ámbitos muy lentos este problema puede ser irrelevante, pero si la gestión de un pulsador sirve, como en este caso, para activar una rutina de interrupción, cualquier microcontrolador consigue contar todos los contactos con consecuencias inimaginables. El debounce consiste en considerar como un solo contacto todos aquellos que se verifican en un intervalo de tiempo: típicamente entre 100 y 200 ms.
El firmware es cargado en el micro mediante la técnica ISP, ampliamente ilustrada en la Guía a la Programación de los microcontroladores ATMEL, descargable gratuitamente del link: *www.micheleMenniti.it/Arduino_burn_bootloader.php*. Es de fundamental importancia ajustar el fusible del micro (ATmega328P) con los siguientes valores: LOW= e0, HIGH=df, EXTENDED= 07, debido a que estos serán calculados para el funcionamiento con el oscilador de cuarzo externo de precisión previsto en nuestro proyecto.

CALIBRACIÓN Y USO DEL FRECUENCIMETRO

Bien, en este punto estamos listos para la fase final: la calibración; se trata de una operación simple pero también extremadamente delicada, debido a que de ella depende la precisión que tendrá vuestro frecuencímetro. Lo ideal sería poder disponer de un generador de señales profesional, pero en realidad es más que suficiente cualquier señal, entre 1 MHz y 5 MHz,

cionados (B, T o R); este último será mostrado alternado con el carácter "*", creando una especie de parpadeo que corresponderá a la frecuencia de entrada, es decir, al ciclo de lectura (1 segundo).

Como hemos visto estudiando la sección hardware, la señal de entrada puede llegar al micro directamente (después de eventuales amplificaciones y/o comformadores de onda), dividida x10 o aún dividida x400. Además hay que considerar el comportamiento del micro que, cuando recibe en la entrada señales con frecuencia superiores a 6 MHz, tiende a dar resultados completamente erróneos e incontrolables; por este motivo se ha marcado un límite de 5,5 MHz, para las entradas BF y TTL, a partir del cual se activa el divisor x10. Entonces cada señal, después de la comprobación de la rutina Checkingr, viene tratada por una routina sucesiva, seleccionada entre chkBF, chkTTL y chkRF, para ser devuelta al valor original, si es necesario. Algún ejemplo aclarar el mecanismo.
Al encender del instrumento, el firmware, por el ajuste predefinido, activa la entrada BF, manteniendo el divisor x10 desactivado e inicia inmediatamente la lectura de cualquier señal aplicada a tal entrada.
Imaginando ahora que se aplica una señal de 1 MHz a la entrada BF (1.000.000), siendo un valor inferior a 5,5 MHz, la frecuencia original llegará al micro y la rutina chkBF no hará ninguna operación dado que no es necesario.
Imaginemos ahora, con el divisor x10 desactivado, que aplicamos una señal de 8 MHz (8.000.000); siendo superior a 5,5 MHz la rutina chkBF activa inmediatamente el divisor x10 y "ordena" una nueva lectura que, obviamente llevará al micro solo 800kHz (800.000); a este punto la rutina multiplicará tale valor x10, llevándola a 8 MHz pero no desactivará el divisor x10, en previsión de mas lecturas de tal señal.
Si ahora aplicamos a la misma entrada BF una señal de 3 MHz (3.000.000) esto, en función del divisor activo llegara al micro con valor 300 kHz (300.000).
En este caso la función chkBF multiplica la lectura x10, restaurando el valor original, entonces desactiva inmediatamente el divisor x10 y ordena una nueva lectura que esta vez, naturalmente, llevará al micro la frecuencia directa (3.000.000).
El comportamiento del proceso para las señales aplicadas a la entrada TTL es idéntico al anterior visto para la entrada BF; solo que siendo el rango de lectura mucho más amplio (recordemos que es posible leer señales hasta los 50 MHz), hemos predispuesto que, cuando se selecciona tal entrada, se active automáticamente el divisor x10. En el caso de señales provenientes de la entrada RF, dado que siempre se dividen x400, la relativa rutina se limita, a cada ciclo de lectura, a multiplicar x400 el valor que llega al micro.

posiblemente de onda cuadrada, porque es estable y precisa. Primero de todo, sin embargo, es necesario regular el contraste del display para obtener la mejor visibilidad: la operación se efectúa girando el potenciómetro R6.
Recordar que, como ya explicamos en la sección relativa al firmware, la retroiluminación es gestionada automáticamente por él, en base a la detección o menos de una señal en la entrada del micro, por eso no hagáis caso a su comportamiento, al menos por ahora.
Antes de comenzar la calibración es necesario plantear correctamente algunas líneas del firmware, como se indica a continuación.

- En la sección "VARIABLES":

```
volatile unsigned val = 0;
```

- En la rutina *lcdprint* es necesario reactivar las líneas:

```
lcd.print(val);      // visualiza la calibración.
lcd.print(" ");
```

- En la rutina loop es necesario reactivar la líneas:

```
for (int i=0; i<media; i++)
        {
         val += analogRead(f_comp_reg);
        }
  val = (val/media); // calculo media y multiplicador ADC
```

Como cada instrumento de medición, también nuestro frecuencímetro necesita alcanzar una temperatura operativa estable, por lo que aconsejamos realizar las operaciones de calibración después de estar encendido al menos 30 minutos.
Os recordamos que la calibración real consiste en encontrar el mejor valor para la constante de calibración *f_comp*, mediante la variable *val*. En nuestras pruebas iniciales nos hemos chocado con el hecho de que cada intento nos obligaba a reprogramar el micro, entonces hemos pensado transformar, pero solo temporalmente, la graduación software en una graduación hardware.
Come ya sabéis, el potenciómetro de precisión R2 (un multivuelta vertical) sirve para aplicar al pin 28 del micro (pin 5 del ADC interno) una tensión variable entre 0V y aquella generada por Aref (1,1 V aproximadamente, para ajuste software); tal tensión, por obra del ADC, se convertirá en un valor entero entre 0 y 1.023, que será asumida por la variable val y después por la constante de calibración *f_comp*. De esta manera, una vez aplicada la frecuencia de muestra a la entrada más idónea para medirla, y seleccionada la escala en Hz (para tener la máxima resolución posible), será necesario girar el cursor del potenciómetro R2 hasta leer sobre el display el valor de la frecuencia de muestra. Aunque el potenciómetro es un multivuelta, la regulación es fina, por lo que cuando comencéis a leer un valor bastante preciso debéis

girar muy delicadamente el cursor; además, y esto es muy importante, considerando que hemos aplicado el método de la media, para estabilizar la lectura del ADC, después de cada retoque del cursor conviene siempre realizar un par de lecturas para ver el efecto correcto. Cuando alcancéis el valor más preciso posible, no toquéis más el potenciómetro y tomar nota del valor visualizado en el display (por ejemplo 992).
En este punto apagar el instrumento y modificar nuevamente el firmware, como se indica a continuación.

- En la sección "VARIABLES" escribir:

```
volatile unsigned val = xxx;
// en el lugar de xxx introducir el valor visualizado
// sobre el display a graduación completada (por ejemplo
992)
```

- En la rutina *lcdprint* es necesario desactivar las líneas:

```
/*
lcd.print(val);    // visualiza la calibración.
lcd.print(" ");
*/
```

- En la rutina *loop* es necesario desactivar las líneas:

```
/*
for (int i=0; i<media; i++)
        {
         val += analogRead(f_comp_reg);
        }
val = (val/media); // calculo media y multiplicador ADC
*/
```

A continuación hay que reprogramar el micro. Desde este momento en adelante el potenciómetro R2 estará desactivado.
Habíamos pensado dejar el método de calibración hardware como definitivo, pero en las varias pruebas realizadas nos hemos dado cuenta que el ADC sufre siempre alguna influencia por parte del circuito, sobre todo en las mediciones RF, pero también en las diferentes condiciones operativas, por lo que el valor tiende a variar de ±1 y esto conlleva variaciones de lectura también importantes, especialmente a frecuencias elevadas. Sin embargo el valor fijado como constante es invariable y no sufre ninguna alteración. Naturalmente, con el tiempo los componentes tenderán a cambiar sus características, por lo que podría hacerse necesaria una nueva calibración; en este caso, ningún problema: bastará con restaurar el firmware y repetir la operación hardware.
No olvidéis nunca que la calibración la habéis realizado después de 30 minutos de encendido del instrumento, porque podréis obtener la máxima precisión solo después de haber encendido el instrumento por ese mismo periodo de tiempo, durante la fase de calentamiento, es fácil leer valores no precisos.
Hablemos ahora de un componente fundamental, pero muchas veces olvidado, en la detección de la medición en frecuencia: la sonda. En el mercado se encuentran los clásicos cables con un extremo BNC macho y por el otro lado los dos cocodrilos rojo y negro; existen también los cables BNC macho–BNC macho, las sondas, a alta o baja impedancia, para mediciones de señales RF o BF. En conclusión, la elección es muy amplia, pero si se realiza mal se pierden las prestaciones del instrumento, tanto en términos de sensibilidad como en términos de máxima frecuencia.
Una sonda construida con el cable común blindado para baja frecuencia proporcionara buenas lecturas de la entrada BF, pero hasta 40÷50 kHz, al máximo 100 kHz, después las prestaciones disminuirán visiblemente. La entrada que causa menos problemas es la TTL, en cuanto las señales tiene amplitudes notables, de 3 a 12 Vpp, cualquier cable apantallado de una cierta calidad permitirá leer también los 50 MHz. Sin embargo la cuestión llega a ser extremadamente problemática con la medición de señales de radiofrecuencia, donde es indispensable recurrir a sondas específicas, construidas con cables que deben ser almenas del tipo RG58; pero este tipo de cables, a partir de 500÷600 MHz provocan una atenuación de la señal que empeora la sensibilidad de lectura del instrumento, sin embargo sin superar el metro de largo el RG58 proporciona prestaciones de optimo

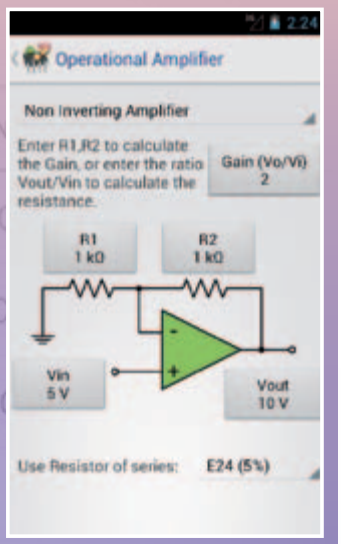
nivel. Como prueba de ello, considerar las sensibilidades indicadas en las características del instrumento ha sido realizados propiamente con este tipo de cable, en la configuración a doble BNC macho, que es la condición ideal, ya que no hay dispersiones.
Pero el sistema del doble BNC solo se utiliza en las mediciones de instrumentos dotados de tal salida; lo ideal para las mediciones RF son las sondas específicas, que terminan con un punta rígida y un cocodrilo para conectar a la masa del circuito bajo test; estas están protegidas para eventuales picos de elevada tensión y permiten tomar medidas en puntos poco accesibles. Las sondas que terminan en doble cocodrilo, generalmente van bien para prototipos, breadboard, base perforada o circuitos con componentes espaciados y sobre los cuales es posible enganchar la borna; en caso contrario pueden también provocar cortocircuitos, por lo que al usarlas es necesario prestar mucha atención.
En cuanto a la protección de las entradas, nuestro frecuencímetro es capaz de soportar amplitudes y potencias bastante elevadas, pero naturalmente el instrumento es usado de manera adecuada, sobretodo en la entrada RF.
Para señales particularmente potentes, tales como aquellas a la salida de transmisores RF, es necesario adoptar un sistema de protección, como un atenuador de señal o una sonda protegida adecuadamente, o se puede evitar el contacto directo entre la sonda y la señal mediante dos técnicas que, naturalmente son eficaces solo con señales de elevada potencia.
Para la primera técnica habíamos previsto una entrada de tipo SMA, debido a que con este tipo de conexión existen en mercado distintos tipos de antenas sintonizadas sobre las frecuencias estándar hoy en uso: 434 MHz, 868 MHz, 915 MHz, 2,4 GHz las más conocidas. Conectando una de estas antenas (en base a la frecuencia a medir) a la toma SMA y activando la trasmisión del aparato RF a medir, si la señal es adecuada será suficiente acercar las dos antenas entre ellas para leer la relativa medida sobre el display, evitando el contacto eléctrico e por lo tanto la sobrecarga de la entrada del frecuencímetro.
Otra técnica, menos eficaz pero extremadamente económica consiste en realizar una antena cableada, el factor fundamental es la longitud del trozo; se calcula mediante una fórmula muy simple:

$l = 299/F$

donde **l** (*lambda*, longitud de onda) es la longitud del cable expresada en metros y **F** la frecuencia a medir, en MHz. Si el valor **l** es excesivo, se puede recurrir al factor lambda, dividiendo sucesivamente el resultado por 2 o 4 o 8, y obteniendo entonces una antena de 1/2, 1/4 o 1/8 de longitud de onda.
Veamos un ejemplo, suponiendo que queramos verificar la correcta trasmisión de un aparato RF de CB (*Citizen Band* o Banda Ciudadana, sobre 27-30 MHz): en general estos aparatos emiten potencias de 2 o también 5 watt, que, si se utiliza la conexión directa, pueden representar un problema para la entrada RF. Recurriendo a la formula anteriormente citada, vemos que nos serviría un cable de largo 299/27=11,07 metros: ¡sin duda un poco grande para tener en un banco de trabajo! Este resultado se puede entonces dividir por 2 (5,5 m), por 4 (2,77 m) o por 8 (1,38 m). Cualquiera de estas longitudes, obviamente realizada con cable de buena calidad, es válida para detectar

La electrónica **según Android**

Entre las innumerables App para el mundo Android no podía faltar una dedicada a la electrónica: ElectroDroid es un conjunto de herramientas (disponible en versión gratuita que conlleva la aparición de banners publicitarios, o de pago, libre de banner) que contienen el código de colores y números para las resistencias normales y SMD, el código de colores de los inductores y las fórmulas para el cálculo de reactancia y resonancia capacitiva / inductiva, del divisor de tensión y de las resistencias y condensadores en serie y paralelo, y el dimensionado de los operacionales, de los reguladores de tensión, de los circuitos que contienen LED, de simples filtros paso bajo/alto, de los circuitos con el NE555 y la distribución de pines de los conectores más comunes utilizados en electrónica (USB, serie, paralelo, Ethernet, RJ, SCART, DVI, HDMI). La App soporta la descarga de plugin para expandir las ya numerosas funciones.



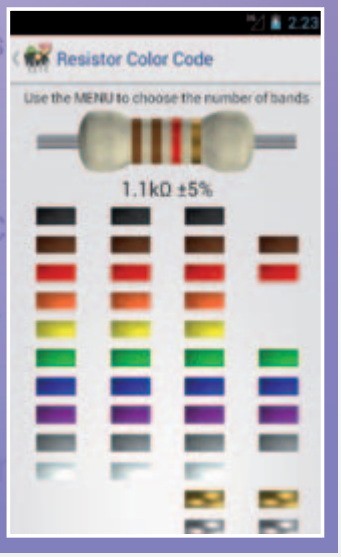


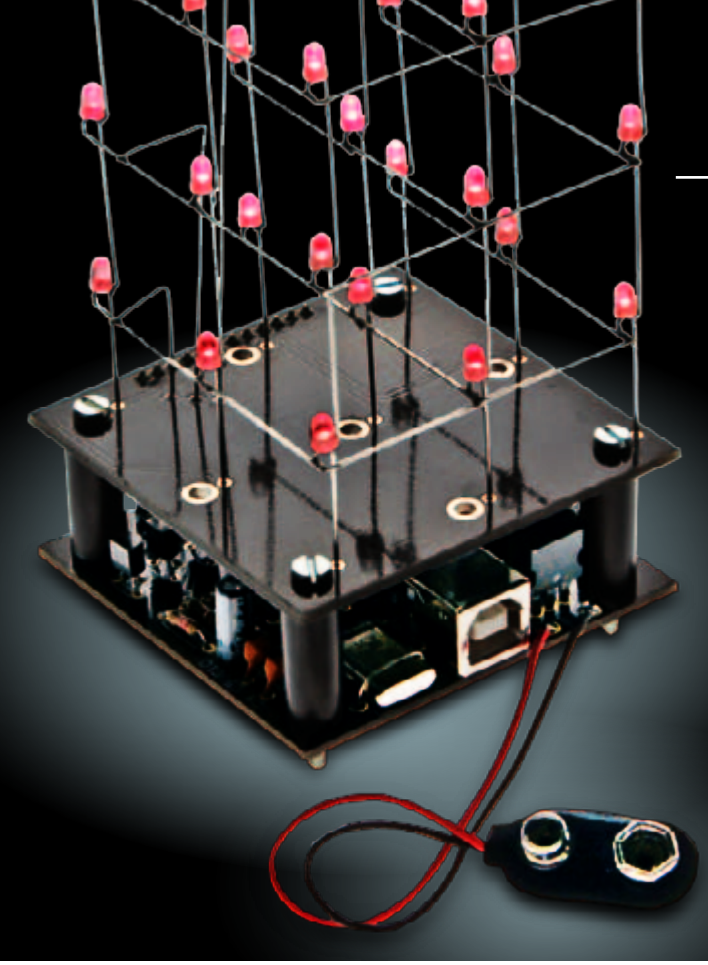
***Fig. 4** - Visualización de la medición de 1GHz.*

señales de frecuencia entre 24 y 30 MHz, teniendo en cuenta que las tolerancias (bien sea en longitud o en frecuencia) son bastante elevadas: de hecho el cálculo se efectúa teniendo en cuenta la frecuencia de centro banda del rango que se desea medir. Pero si en un extremo del cable se crimpa con una conector BNC macho, ¿que se pondrá en el extremo opuesto? Bueno, se puede dejar libre, simplemente quitando la cubierta algún centímetro, o entre central y la cubierta se puede soldar un condensador cerámico, de manera que se crea una especie de arco que captará la RF permitiendo a la entrada específica amplificar la amplitud y medir la frecuencia.
Muchos realizan estas antenas de hilo con un cable normal de cobre con cubierta, especialmente si son muy largas, para poder así recogerlo después, obteniendo buenos resultados. Una vez que tangas entre las manos el frecuencímetro listo y funcionando podréis hacer todas las pruebas que queráis, hasta encontrar aquellas más adecuadas a vuestras exigencias.
Ahora, un rápido resumen sobre el simple uso del instrumento: al encenderlo se seleccionaran automáticamente la entrada BF (LED verde) y el rango de medida "Hz"; el display LCD, en ausencia de señal tendrá una retroiluminación muy baja.
En estas condiciones es necesario contar con 20-30 minutos de precalentamiento (o tendrás que contar con una mínima imprecisión inicial en las lecturas), después se puede seleccionar la entrada deseada (si es propiamente el BF no es necesario hacer nada), mientras a lo que se refiere a la escala, no hay problema, podéis recorrer a placer todos los rangos (Hz, kHz, MHz y GHz) sin miedo de over-range (fuera de rango), en cuanto al número de cifras activas es abundante también sobre el rango más bajo (Hz).
Cuando cambia el rango, el display muestra automáticamente no solo la unidad de medida, sino también un punto separador (excepto rango en Hz) para facilitar la lectura de las cifras.
En la **Fig. 4** el display visualiza bien 1.000 MHz, obviamente detectado sobre la entrada RF; tener en cuenta, de hecho, la indicación de rango de medida (MHz), el punto separador entre la parte de los MHz y la restante parte del valor visualizado, y la letra "R" que indica la entrada activa en este momento.
Nota final: considerar que, como todo instrumento de medida, también nuestro frecuencímetro garantiza la lectura ±1 digito, es decir, es normal que la última cifra tienda a cambiar en una unidad; por ejemplo, la lectura de una frecuencia de 1.258 Hz podría variar entre 1.257 y 1.259 Hz. Pero en nuestro caso es necesario considerar el factor divisor automático; para poder suplir a la problemática de la frecuencia máxima que consigue leer el micro hemos recurrido a dos divisores: uno, fijo, que divide cada señal RF x400 y otro, automático, que si es necesario puede dividir la señal BF o TTL x10. En estos casos el "digito" se representa justamente por estos factores de división. Es decir que mientras una frecuencia BF o TTL leída en ausencia de divisor (entonces el máximo es 5,5 MHz) tiene una resolución de ±1Hz, una frecuencia mayor de 5,5 MHz (siempre sobre BF o TTL) que necesita rápida conexión en automático del divisor x10, tendrá una resolución de ±10Hz; por ejemplo 8,5 MHz leídos en escala Hz se convierten en 8500000, pero en este caso el "digito" se convierte en la segunda cifra desde la derecha, y no la primera, entonces podríais leer 8499990 o 8500000 o 8500010.
De la misma manera, en las mediciones RF, en las cuales tenemos un divisor fijo x400, la resolución se convierten justamente de ±400Hz; por ejemplo leyendo 550 MHz, el "digito" se convierte en la tercera, entonces podríais ver 549999600 o 550000000 o 550000400.
De cualquier forma, un "error" sobre la 6ª cifra en el orden de MHz o sobre la 7ª cifra en el orden de las centenas de MHz es bastante insignificante como para poder ignorarlo; es necesario de hecho considerar que muchos generadores de señal tienen incluso una resolución más baja, no llegando más allá de las seis cifras completas.

(180033)

el MATERIAL

Todos los componentes utilizados en este proyecto son fáciles de encontrar. Los originales de los circuitos impresos, así como el firmware utilizado para programar el microcontrolador Atmel, se pueden descargar de la web de la revista.
¡Conéctate enseguida a www.nuevaelectronica.com!



CUBO LUMINOSO

Realizamos un cubo constituido por tres planos y 9 columnas de LED, manejado por un microcontrolador que le permite generar muchos efectos luminosos sorprendentes, tanto memorizados, como enviados desde un PC conectado mediante USB.

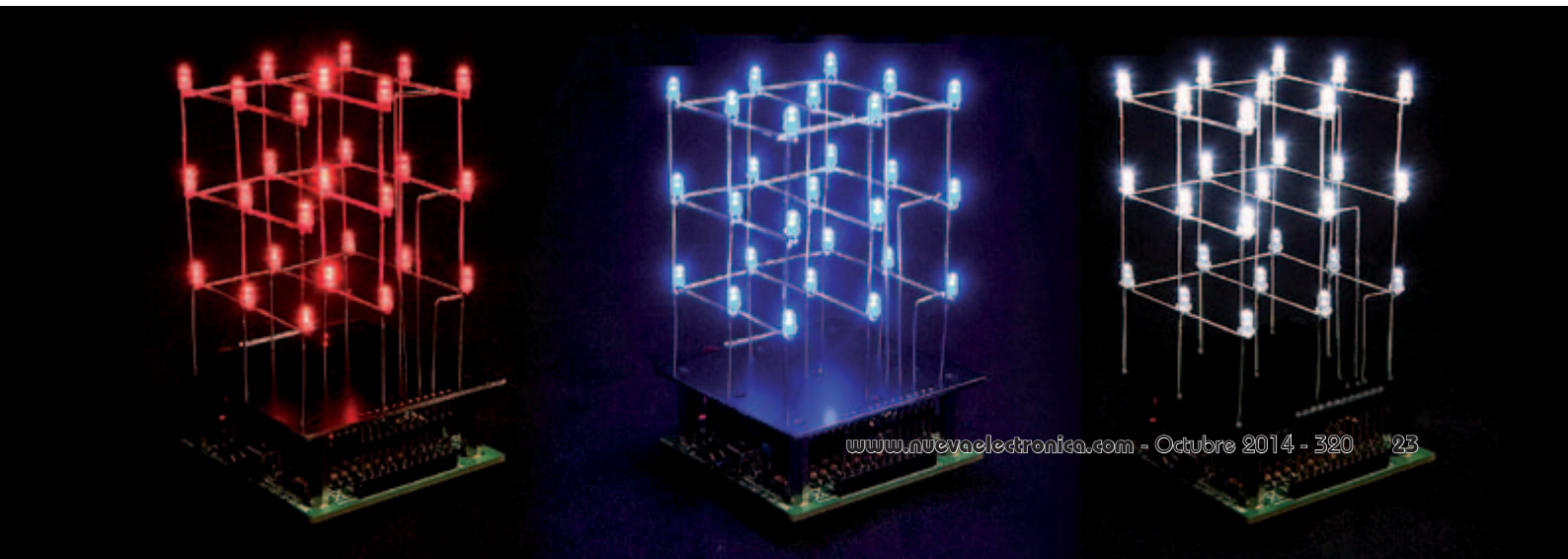
DAVIDE SCULLINO

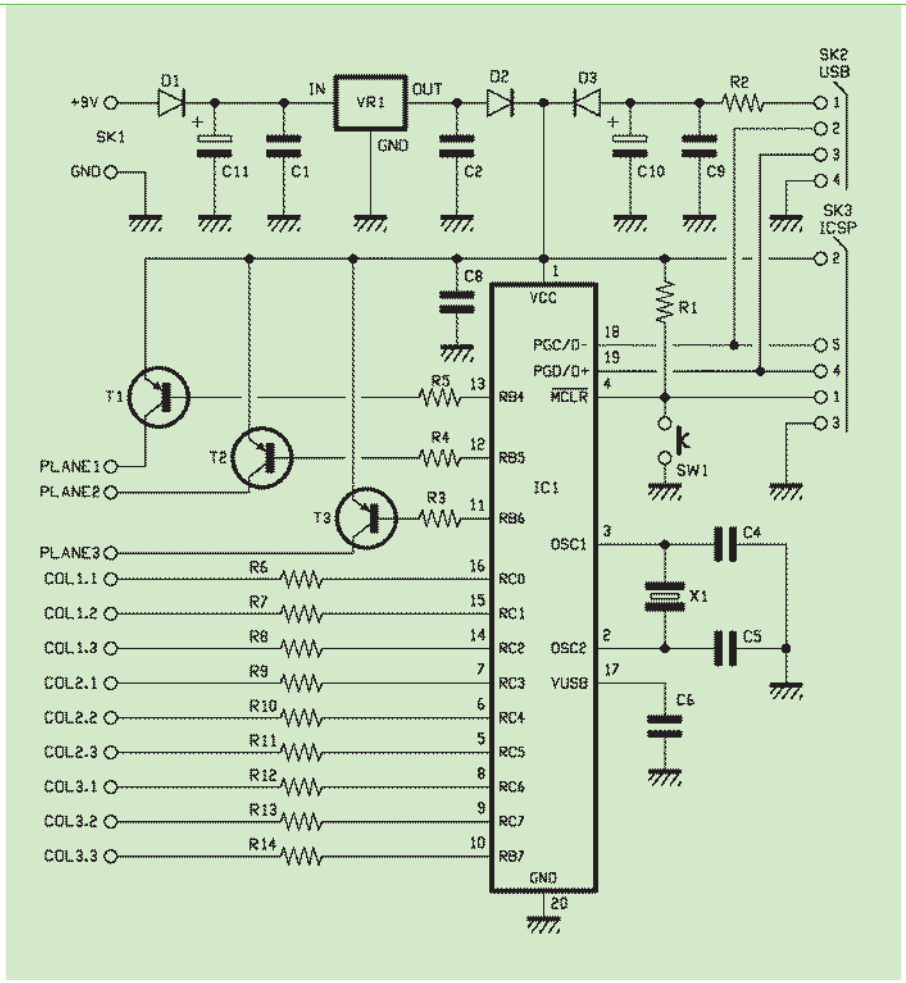
El cubo de LED es uno de los efectos luminosos más cautivadores, ya que permite materializar juegos de luces tridimensionales y no simplemente planos, como aquellos obtenidos con un circuito impreso cubierto de LEDs; claramente, realizar un circuito así no es nada simple, en cuanto el hardware consta de una estructura tridimensional compuesta por planos y columnas de LED, a manejar sincronizadas (normalmente se recurre a la técnica de multiplexado) a alta velocidad, cuanto más complejo es la figura 3D elegida más alta es la velocidad de ejecución, se trata de efectos que proponen figuras en movimiento desde arriba hacia abajo, de izquierda a derecha y de delante hacia detrás (estamos hablando de un sólido, que típicamente es un cubo). Por esta razón los controladores de este tipo están generalmente basados en un microcontrolador: no es casualidad ver estos juegos de luces con Arduino, por ejemplo.

Lo que os proponemos aquí es un cubo luminoso con 3 planos y 9 columnas (27 diodos luminosos en total) manejado por un microcontrolador y provisto de interfaz USB para permitir a un ordenador personal, en el que funciona un software específico, ordenar los efectos luminosos a través de una pantalla de control simple e intuitiva. El software se puede descargar gratuitamente de nuestra web

ESQUEMA ELÉCTRICO

Vayamos enseguida a echar un vistazo al circuito electrónico del cubo: el esquema eléctrico visible en estas páginas muestra la tarjeta de control excluidos los LEDs, los cuales están montados según un dibujo que realiza el cubo luminoso, del que hablaremos más adelante.





El circuito está construido alrededor del microcontrolador IC1, un PIC18F14K50-I/P de Microchip, que se conecta con el ordenador que ejecuta los juegos de luces ordenando en multiplex tres líneas para activar los tres planos del cubo, y 9 I/O para gestionar las columnas de LED dispuestos en vertical, tres por plano.
El PIC18F14K50-I/P es un microcontrolador dotado de interfaz USB de tipo device (periférico) que funciona a la velocidad Full Speed 2.0 del USB y el dialogo sobre el bus es gestionado a la velocidad que marca el reloj, programado a su vez por el cuarzo X1. Este cuarzo define también la frecuencia de funcionamiento de la CPU y de las etapas internas del IC1.
El USB está localizado en los pines 18 (D-) y 19 (D+), compartidos respectivamente con las funciones PGC y PGD del puerto ICSP, activo durante la programación in-circuit del microcontrolador; el ICSP usa también el pin 4, que corresponde al /MCLR, el reset a realizar mediante el pulsador SW1 cuando es necesario iniciar la programación. Esta línea normalmente debe quedarse a uno lógico (es activa a nivel bajo) y por este motivo en el circuito está presente la resistencia R1, que hace de pull-up. Al conector ICSP llega también la línea de alimentación, compuesta por hilos +5V y masa.
Respecto al tema de la alimentación, analicemos el esquema correspondiente partiendo de una consideración: la lógica y las líneas de control del cubo de LED se pueden alimentar tanto por la alimentación que llega de SK1 (toma móvil para pilas, a la que se puede conectar también un alimentador de red), como por la toma USB (SK2).
La tensión sumiistrada a SK1 atraviesa el diodo de protección de polaridad inversa (D1) en cuyo cátodo se encuentran los condensadores C1 y C11, que filtran la alimentación a la entrada del regulador integrado VR1; este último es un común 7805 en un encapsulado TO-220 que entrega 5 volt bien estabilizado con los que, a través el diodo D2, alimenta el resto del circuito, el microcontrolador y las líneas de alimentación de los planos de LED. En cuanto al USB, la alimentación sobre el contacto +V del conector, oportunamente filtrada por los condensadores C9 y C10, llega al ánodo del diodo y pasa al cátodo, desde el cual llega también a la alimentación del microcontrolador y de los driver de los planos de LED. Los diodos D2 y D3, en definitiva, se comportan como una puerta lógica OR y permiten alimentar el microcontrolador por USB o por alimentador externo sin que una de las fuentes de alimentación se descargue sobre la otra; el micro y el resto de la lógica absorben corriente desde la fuente que tiene mayor potencial en el nodo de unión de los cátodos de los dos diodos.
La resistencia R2, en serie el pin 1 del conector USB tiene la doble misión de proteger el ordenador de posibles cortocircuitos que pudieran ocurrir sobre la línea de alimentación positiva, o en todo caso en el interior del circuito, y contribuir a filtrar los 5 voltios recibidos del USB ya que forma con C9 y C10, un filtro paso-bajo que atenúa los picos presentes sobre los 5 V del USB y por tanto sobre la línea del ordenador. R2 tiene un valor de 10 ohm y siendo 5 voltios la tensión del USB, limita a 500 mA la absorción en cortocircuito; y como cualquier USB de un PC standard llega hasta 500 mA, la resistencia evita en cualquier caso la sobrecarga del PC.
Bien, descrita la etapa de alimentación podemos pasar a examinar el control de los LED: como ya hemos mencionado, las columnas vienen activadas directamente por líneas de I/O del microcontrolador, mientras los planos son controlados por tran-

sistores (NPN) controlados por otras tres líneas del IC1. En esencia, los planos corresponden a la unión de los cátodos de los LED puestos sobre el mismo nivel: 9 por plano. Las columnas son, sin embargo, conectadas a tres ánodos de los 3 LED que constituyen una fila vertical de LED.
Este modo de control nace de la consideración que las líneas de I/O del microcontrolador, usadas como salida, en el modo source cuando pasan a nivel alto deben suministrar corriente a la carga) registran una caída de tensión mayor que la correspondiente a cuando funcionan en modo sink (absorben corriente).
Así, las I/O que deben suministrar corriente (suministrar la alimentación positiva para encender los LED) manejan cada una un transistor PNP, aplicando un cero lógico a la base de manera que el colector entrega la corriente a los LED de plano (son nueve), mientras aquellos que deben absorber corriente (para cerrar a masa los cátodos) son conectados directamente a los LED correspondientes (tres a la vez).
Retomando el esquema eléctrico vemos que RC0 controla la primera columna, RC1 la segunda y RC2 la tercera, RC3 gestiona la cuarta, RC4 la quinta, RC5 la sexta, RC6 la séptima, RC7 la octava y RB7 la novena.
Las columnas son comunes a cada plano, por tanto cada una corresponde a tres LED (plano 1°, 2° y 3°).
El manejo de los LED se realiza, para ahorrar líneas de I/O, en multiplex, en el sentido que la apropiada rutina de encendido prevé un análisis completo en secuencia desde el primero hasta el vigésimo séptimo LED, uno solo cada vez. Empiezaa desde el primer LED del primer plano y se termina con el último del tercero.
Por ejemplo, para encender el primer LED del primer plano llevamos la línea RB4 a nivel lógico bajo y simultáneamente llevamos RC0 a cero lógico y se mantienen a 1 lógico todas las otras líneas de la puerta C, y la línea RB7. Para encender después el segundo LED del primer plano se pone RB5 a nivel bajo de manera que el colector de T2 tenga tensión (los otros dos transistores quedan apagados) y se pone RC1 a nivel bajo y se dejan a 1 lógico las otras líneas del RC y la RB7.
Para que todo funcione, el cubo se ensambla de la manera que se muestra en el diseño de montaje específico.
Debemos decir además que los ánodos se gobiernan con transistores NPN y los cátodos por las respectivas salidas conectadas en modo sink.

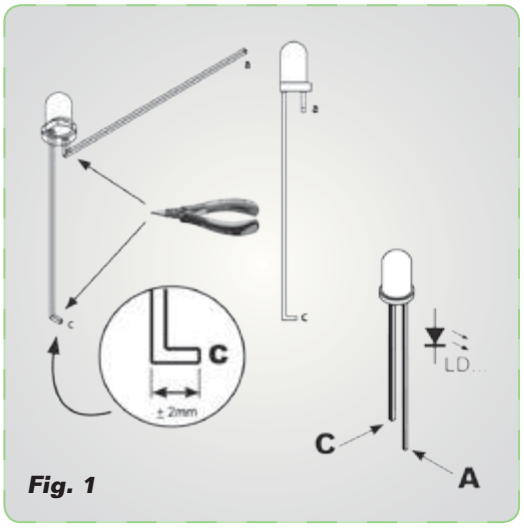


Fig. 1

DESARROLLO PRÁCTICO

Bien, después de ver el funcionamiento del circuito afrontamos la parte más interesante y con un mínimo de dificultad: la construcción tanto del circuito impreso base (que contiene el microcontrolador y todo lo que aparece en el esquema eléctrico), como del que soporta el cubo de LED; este último deberéis montarlo "aéreo" siguiendo nuestras indicaciones.
Empezamos desde el circuito impreso de la tarjeta de control, que se realizará a partir del diseño de las pistas (es doble cara...); una vez grabado y perforado, distribuir los componentes en orden de altura, respetando la posición de aquellos que están polarizados, tal como indica el plano de montaje correspondiente. Completar el circuito montando el conector USB-B sobre el circuito impreso y la toma móvil para la pila de 9 voltios; esta servirá solo si usáis el circuito en modo stand-alone (por ejemplo como gadget de muestra) después de haber cargado en la EEPROM del PIC las animaciones a desarrollar.
Cuando conectéis el dispositivo al ordenador, será el USB de este último quien proporcionará la alimentación, por lo tanto no hacen falta el toma como la pila. Pasar ahora a la construcción del cubo, empezando por el circuito impreso, también

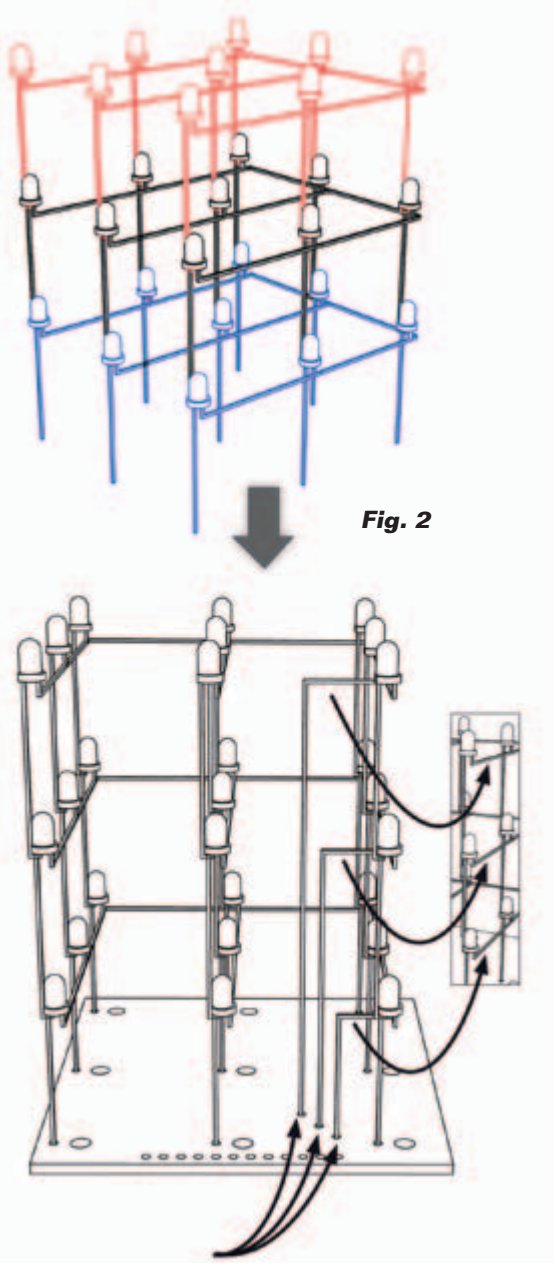
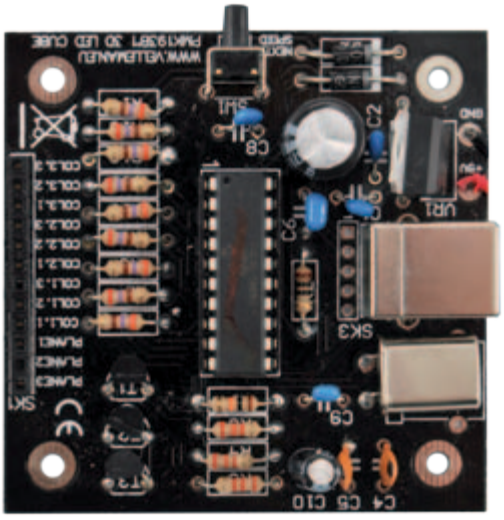


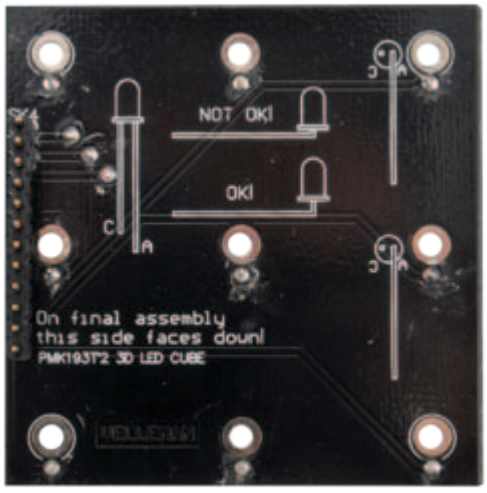
Fig. 2

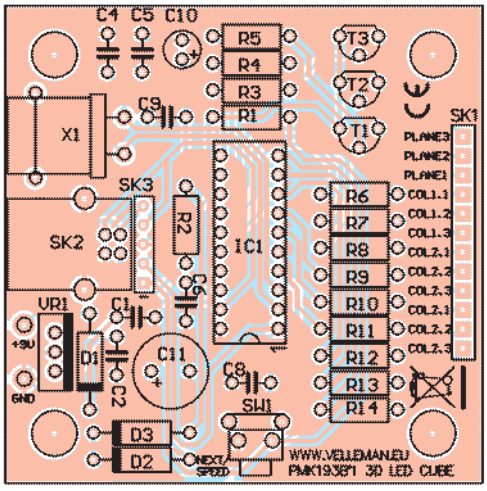
Llsta de materiales

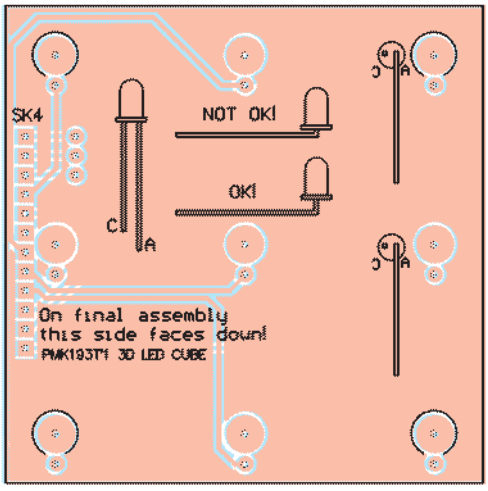
R1: 10 kohm
R2: 10 ohm
R3÷R5: 2,2 kohm
R6÷R14: 270 ohm
C1: 100 nF cerámico
C2: 100 nF cerámico
C4, C5: 22 pF cerámico
C6: 470 nF cerámico
C8, C9: 100 nF cerámico
C10: 4,7 µF/63 V electrolítico
C10: 470 µF/16 V electrolítico
D1÷D3: 1N4007
SW1: Microswitch 90°
T1÷T3: BC557B
X1: Cuarzo 12 MHz
IC1: PIC18F14K50-I/P (software VMK193)
VR1: 7805

Varios:
- LED 3 mm (27 pz.)
- Conector USB-B
- Zócalo 10+10
- Tira de 12 pines hembra
- Tira de 12 pines macho
- Distanciador plástico (4 pz.)
- Tuerca 3 MA (4 pz.)
- Tornillo 3 MA (4 pz.)
- Clip para batería 9V
- Circuito impreso PMK193B
- Circuito impreso PMK193T









a doble cara, que hará de soporte. Obtenido el circuito impreso, soldar un una tira de 12 pines macho de 20 mm largo y dejar a un lado el circuito para centrarnos en la composición del cubo, para lo que puede seros útil observar las ilustraciones **Fig. 1** y **Fig. 2**; en fin, sugerimos montar el primer plano, después el segundo y por último el tercero, distribuyendo los LEDs como se muestran en las mencionadas ilustraciones.
Cada diodo está dispuesto con el cátodo recto el ánodo doblado; el primero va conectado al cátodo del LED que está debajo o arriba (según el plano) mientras el ánodo de los LED de cada plano va conectado a los otros del mismo plano es llevado a la base de la estructura mediante una pieza de hilo de cobre rígido, de diámetro inferior a 1 mm de manera que pueda entrar en la base correspondiente del circuito impreso que soporta el cubo. Ensamblados todos los planos uno encima del otro, colocar los cátodos de los LED de las columnas del plano más bajo en sus respectivos taladros del circuito impreso y los terminales de los cables comunes de los ánodos de los planos en los taladros previstos para ellos. Soldar

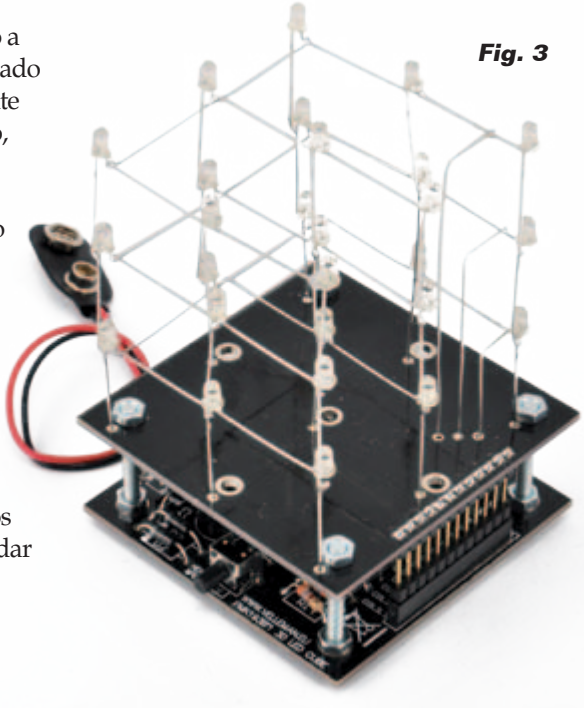


Fig. 3

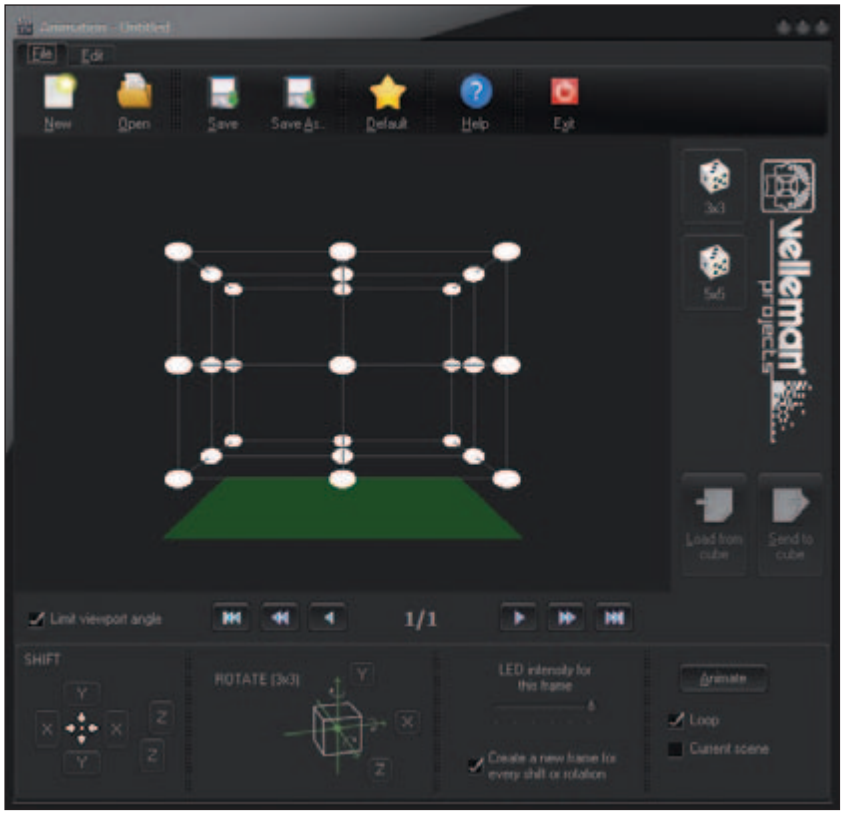


***Fig. 4** - Pantalla de trabajo del software Cube Animator 1.4.*

las tres trozos de cable de la manera indicada en las imágenes que veis en estas páginas (**Fig. 3**), insertar después la estructura en la tira de pines hembra de la tarjeta base. Conectar el sistema a la entrada USB de un PC en el que haya sido instalado el apropiado software de gestión del cubo (se descarga desde nuestra web, www.nuevaelectronica.com, junto con los demás archivos del proyecto). Encender el PC y abrir la ventana de trabajo del software; los LED del cubo empezaran a encenderse. Antes de instalar el software debéis cargar, dependiendo del sistema operativo, el driver para periféricos, proporcionado por Velleman; sin haber instalado este driver el circuito no será leído por el software.

El software de gestión se llama Cube Animator 1.4 (se descarga tanto en nuestra web, como de *www.velleman.eu/support/downloads/?code=MK193*) y, una vez instalado y arrancado, muestra la pantalla de la **Fig. 4**.

Una vez abierto el software, hay que decidir que animación elegir y después prepararos para el envió al USB del cubo; para iniciar la descarga en la memoria del PIC, presionad el pulsador SW1 sobre el circuito y este atenderá la ejecución de las operaciones. Desde el mismo software, accediendo al menú *File* y a través del comando *New* (los comandos de File se representan con sus iconos específicos), se pueden construir nuevas animaciones, apoyándose en la imagen del centro de la ventana, que simula la secuencia de encendido de los LED. Una vez creada la animación, se debe guardar con el comando *Save* del menú *File* o haciendo clic sobre el icono *Save*.

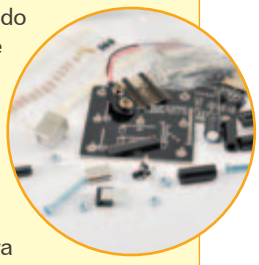
El software permite también abrir archivos de animaciones ya creados de manera que se puedan modificar (usar los comandos del menú *Edit*) y guardarlos con su nombre o con un nombre y un destino distinto (comando o icono *Save As...*). Las animaciones ya creadas pueden ser giradas o desplazadas mediante comandos específicos en la parte inferior de la ventana; en la misma zona está colocado un cursor desde donde podéis regular la luminosidad que los LED tendrán durante la ejecución de la animación actual (es decir aquella que tenemos abierta, sobre la que se está trabajando con el programa).

En fin, a la derecha de la ventana de trabajo de Cube Animator se encuentran los pulsadores *Load From cube* y *Send to cube*: haciendo clic sobre el primero se carga y se abre la animación memorizada actualmente en la EEPROM del microcontrolador del circuito, mientras el segundo sirve para enviar al circuito y memorizar en él, la animación actualmente abierta en el programa para que pueda ser cargada cada vez que queráis.

Ultima nota: cuando debáis crear una animación, hacer primero clic en el pulsador (en encuentra arriba a la derecha de la pantalla) que muestra el cubo que tiene debajo escrito 3x3; de hecho el programa está adaptado también para circuitos con lado de 5 LED, pero en nuestro caso funciona solo la estructura de 3x3 y ajustando el software sobre 5x5 los programas de animación funcionarían mal y crearían problemas a nuestro cubo.

(186069)

el MATERIAL



Este proyecto ha sido producido en caja de montaje por Velleman y esta disponible en la página web de Nueva Electrónica. El kit completo con todos los componentes, microcontrolador ya programado, placa de circuito impreso serigrafiada, y las piezas pequeñas cuesta 28,00 € o (cod. MK193).

Precios IVA incluido sin gastos de envío.
Puede hacer su pedido en:
www.nuevaelectronica.com
pedidos@nuevaelectronica.com

Presentado en Francia eFan, el avión eléctrico di Airbus

El avión eléctrico eFan de Airbus tuvo su primera aparición públicaen Francia en el aeropuerto de Bordeaux-Merignac, ante la presencia del ministro francés de Economía, Arnaud Montebourg.
El avión de adiestramiento ligero del consorcio aeronáutico francés, de emisiones cero, está construido con materiales compuestos, muy ligeros y resistentes y está dotado de dos motores eléctricos.
Los test realizados, como confirmó el piloto de prueba, Didier Esteyne, han demostrado que el vuelo eléctrico ya no es solo una posibilidad hipotética, llegando ya a niveles de rendimiento y eficiencia muy elevados, con el respeto más absoluto del medio ambiente.
Para alimentar los motores del eFan piensan en una serie de baterías de polímero de litio de 250 volt, que permiten al avión permanecer en vuelo durante un tiempo que varía entre 45 minutos y una hora.
Desde hace tiempo Airbus está comprometida en proyectos destinados a garantizar la disminución del impacto ambiental de sus propios aviones como es caso del programa "Flightpath 2050", que tiene el objetivo de reducir el 75% las emisiones de CO_2 y el 65% el ruido de los aviones.
El eFan pronto podría ser utilizado en las escuelas de vuelo para adiestrar pilotos, este es uno de los deseos de Francis Deborde, manager de Acs, la empresa que produce las piezas compuestas del avión.

www.airbus-group.com

Carreteras inteligentes con energía solar: los Estados Unidos lo prueban

Desde Estados Unidos llega el sistema Solar Roadways, enfocado en reproyectar las autopistas americanas haciéndolas más seguras y ecosostenibles, gracias ala sustitución del asfalto común de carretera por verdaderos paneles solares.
La idea base del proyecto, que recibió una primera financiación por parte del Departamento de Transportes estadunidense, consta de la realización de leds solares luminosos incorporados enla superficie de la carretera, capaces, por ejemplo, de señalar cualquier incidente o interrupciones del tráfico durante la noche o esbozar las líneas de delimitación de la carretera. Sin considerar también el ahorro en términos energéticos que supondría poder

PlanetSolar, la campaña 2014 del catamarán solar más grande del mundo

El pasado mes de Abril se inición oficialmente la campaña 2014 del MS Tûranor PlanetSolar, la embarcación solar más grande del mundo con salida desde Lorient en Francia, para amarrar en Boulogne-sur-Mer.
El catamarán solar, invitado de honor en el Festival de imágenes del Mar, promovió en colaboración con el Centro Nacional del Mar, Nausicaa, muchas actividades ligadas a la sensibilización sobre la energía solar y fotovoltaica, con el fin de educar de manera adecada al público sobre los temas ambientales de mayor interés para los ciudadanos. En particular, se han organizado visitas para los estudiantes y la proyección del documental sobre la expedición científicacapaz de estudiar todos los últimos descubrimientos sobre la Corriente del Golfo.
Durante el desplazamiento entre Lorient y Boulogne-sur-Mer, la embarcación atravesó sin problemas las 420 millas náuticas que separan las dos ciudades, respetando los tiempos previstosen el programa.
La MS Tûranor PlanetSolar continuó su viaje en el Mar Mediterráneo atracando en Attalayoun en Marruecos, donde las autoridades locales están muy interesadas en invertir en energías renovables para revalorizar la zona, paradespués virar hacia el Principado de Mónaco y ser el invitado de honor del "Monte Carlo Solar1 Cup 2014", la primera carrera de barcos solares deMónaco.

www.planetsolar.org

disponer de una fuente de energía limpia e ilimitada como es la solar, capaz de liberar del uso de lámparas eléctricas caras y obsoletas, que pueblan las carreteras estadounidenses.
Además, hay que destacar que las superficies de leds utilizadas en el proyecto son muy resistentes, tienen una duración efectiva de incluso veinte años y aún así, en caso de desgaste, los componentes individuales estropeados son fácil sustitución.
Adía de hoy, el marimonio Brusaw, artífices del proyecto, han conseguido realizar un primer prototipo funcional del Solar Roadways en el Nortede Idaho, que ha obtenido óptimos resultados y ha sido certificado con el mismo "agarre" que las superficies asfaltadas. Pero ahora se mira mucho más alto, es decir para iniciar la producción a gran escalay para ello es necesario un millón de dólares, que según los deseos de los promotores del proyecto, se recuperaran gracias a la campaña de crowfounding llevada a cabo en el portal indiegogo.com.

www.solarroadways.com

Bike Intermodal: una start up italiana para el proyecto de la bici plegable

Pocas cosas se parecen al simple placer de una pedalada, pero el uso de la bicicleta puede ser realmente complicado en áreas urbanas con tráfico, donde los viajeros tienen que atravesar a menudo la ciudad de una punta a otra y hacer frente al constante aumento de robos. Han facilitado la movilidad las bicicletas plegables, cada vez más extendidas gracias a la integración con el coche y el transporte público.
El proyecto Bike Intermodal, financiado por la UE, ha desarrollado un nuevo prototipo de bicicleta plegable, que pesa solo 7,5 kg y puede colocarse en una caja de apenas 50 x 40 x 15 cm, fácil de guardar en casa, trasportarla o simplemente apoyarla en la esquina de un restaurante, un bar o un cine. Un motor diseñado ad hoc por Maxon Motor potencia la movilidad sin añadir un peso excesivo. Aún con el motor, la bicicleta pesa de hecho la mitad de un modelo plegable similar pero sin motor.

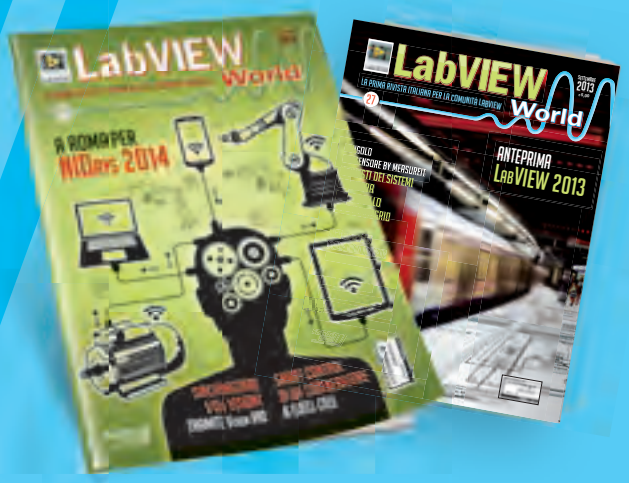
www.bike-intermodal.eu

Los trenes holandeses se alimentaran por energía eólica

Cambio para los transportes ferroviarios en Holanda: desde 2015 los convoys de la red ferroviaria nacional serán alimentados porenergía eólica. La utilidad Eneco se encargará de proporcionar a los convoys 1,4 TWh de energía eléctrica necesaria, cogiéndola de las plantas eólicas nacionales y explotando las instalaciones presentes en Bélgica y Escandinavia.
De acuerdo con los planes, para el 2015 la mitad de los trenes eléctricos holandeses serán alimentados con energía eólica, llegando al 95% en el 2017 y en el transcurso de 4 años la red ferroviaria holandesa completa podría funcionar totalmente gracias a la energía limpia. El acuerdo ha sido firmado por la cooperativa Vivens, una joint venture para la adquisición de energía que incluye a Netherlands Railways (NS), Veolia, Arriva, Connexxion y operadores del transporte de mercancías con ferrocarril. Los holandeses están muy sensibilizados con los temas ambientales como se confirma en un reciente sondeo realizado por Netherlands Railways, que evidencia como 8 pasajeros de cada 10 consideran muy importante reducir el impacto ambiental durante los viajes en tren, acogiendo el uso cada vez mayor de las energías renovables en el sector de los transportes.
Los consumos de energía por kilómetro y por pasajero han sido ya reducidos en un 30% por Netherlands Railways y con el nuevo acuerdo, las emisiones de dióxido de carbono producidas pasaran de 30 gramos a 0 por cada uno de los 1,2 millones de pasajeros diarios.
La iniciativa permitirá a Holanda cumplir con el objetivo en 2020 de producir el 14% de sus necesidades energéticas usando las energías renovables, para después llegar a la cuota de los 4,45 GW en el 2023.

http://news.eneco.com

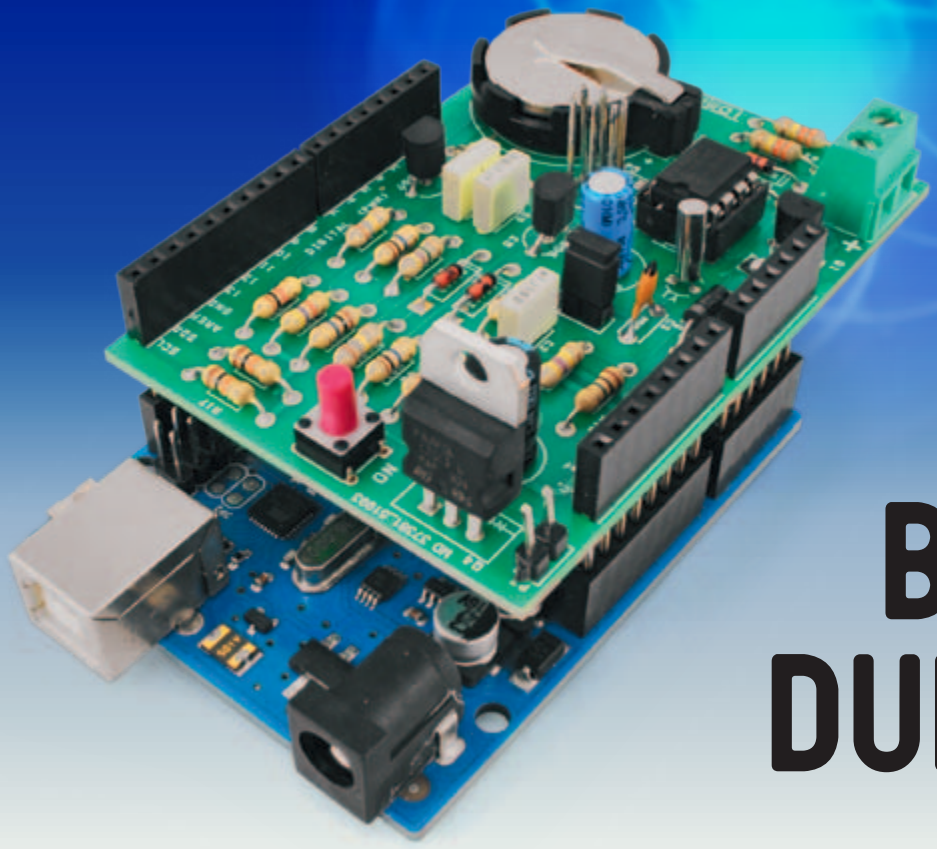
Nuestra *misión* es la ***divulgación de la electrónica***



WWW.OPEN-ELECTRONICS.ORG

WWW.ELETTRONICAIN.IT

WWW.FUTURASHOP.IT

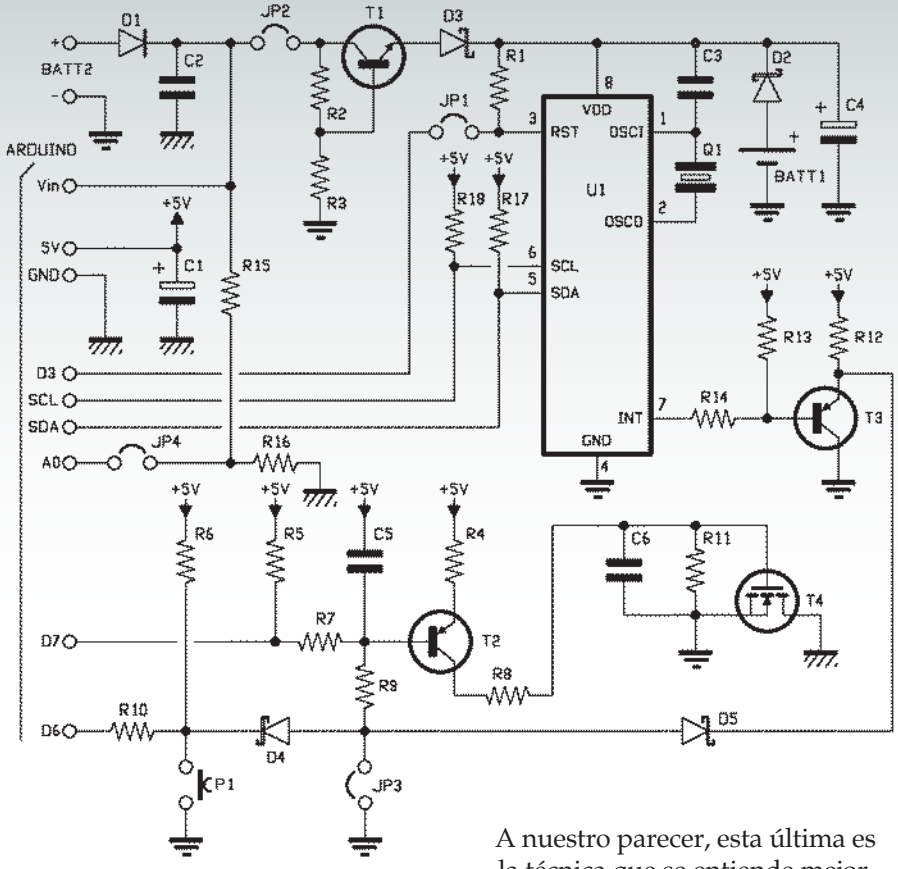


Optimizamos la autonomía de los sistemas Arduino alimentados por batería, gracias a un temporizador basado en RTC con función despertador programable.

ARDUINO BATERÍA DURADERA

Ing. TOMMASO y ALESSANDRO GIUSTO

Como alrededor de Arduino han nacido rápidamente tantas librerías y hardware, aquello que falta en el panorama de la tarjeta del equipo Arduino es una solución cualitativamente válida para optimizar el consumo eléctrico cuando el sistema se alimente por batería. Es cierto que existen técnicas software que permiten limitar la frecuencia del reloj principal de manera que disminuya la absorción de corriente pero desde nuestro punto de vista no son capaces de garantizar una duración aceptable de la batería. Hasta ahora la alimentación por batería para Arduino se ha pensado siempre para hacer frente a breves y temporales pérdidas de la red eléctrica, y nunca como fuente principal de energía; también porque la alimentación de los sistemas electrónicos por baterías es una aplicación bastante crítica y requiere de técnicas especiales combinadas de hardware/software para limitar el consumo de energía. En general en una tarjeta con microprocesador o microcontrolador, el componente que requiere más



energía es seguramente la CPU; de esta consideración han nacido técnicas que aprovechan al máximo la capacidad de las modernas MCU para trabajar al 100 % solo por el tiempo necesario para procesar los datos de entrada/salida, y de ponerse (a través de instrucciones específicas) en estado de "sleep" o bajo consumo (las secciones no necesarias del micro, como los registros de E/S, los timer y los módulos PWM se apagan) por el tiempo restante. Cuando esto no basta, para optimizar el consumo y aumentar la autonomía dada por la batería, se puede recurrir a soluciones que apagan completamente el sistema y lo vuelven a encender periódicamente o cuando lo requiere un evento externo; en la práctica, se trata de utilizar interruptores temporizados o controlados por circuitos o eventos.

A nuestro parecer, esta última es la técnica que se entiende mejor para utilizarse en sistemas basados en Arduino y que ofrece las mejores prestaciones con respecto al tiempo de duración de la carga de la batería.

NUESTRA SOLUCIÓN

El shield presentado en estas páginas es algo de este estilo: se trata de un temporizador con Reloj de Tiempo Real (RTC o *Real Time Clock*) programable por Arduino capaz de, cuando este se apaga, despertarlo el día y hora prefijados. Específicamente, el shield está pensado para ser combinado con las tarjetas Arduino Uno. Está diseñado para recibir (a través de un terminal) la tensión de alimentación proveniente de una batería (son soportadas tensiones de alimentación comprendidas entre 5 y 12 voltios) y, aprovechando una particular funcionalidad del chip PCF8593T (un reloj/calendario), que permite encender el sistema a intervalos fijos seleccionables por software según las condiciones encontradas.

Como veremos luego con más detalle, el hardware del shield permite al software Arduino elegir cuando "ser despertado" y, una vez encendido, a través del estado de una salida digital, por cuanto tiempo permanecer encendido e incluso cuando apagarse.

Resumiendo, se puede decir que la técnica general de funcionamiento es la siguiente (**Fig. 1**):

- el sistema se apaga por un determinado intervalo de tiempo;
- en un cierto momento el reloj despierta a Arduino; este último decide permanecer despierto y pasa a analizar el propio estado interno, así como el de cualquier pin/sensores externos;
- terminado el análisis de las entradas y realizada la actuali-

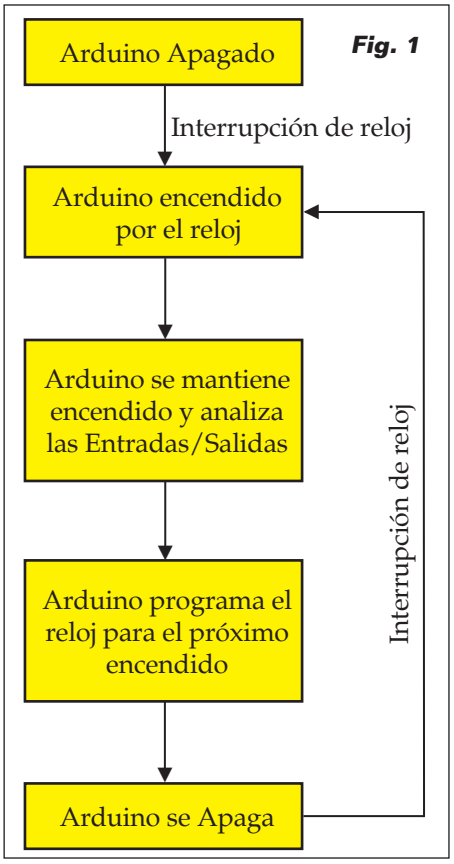


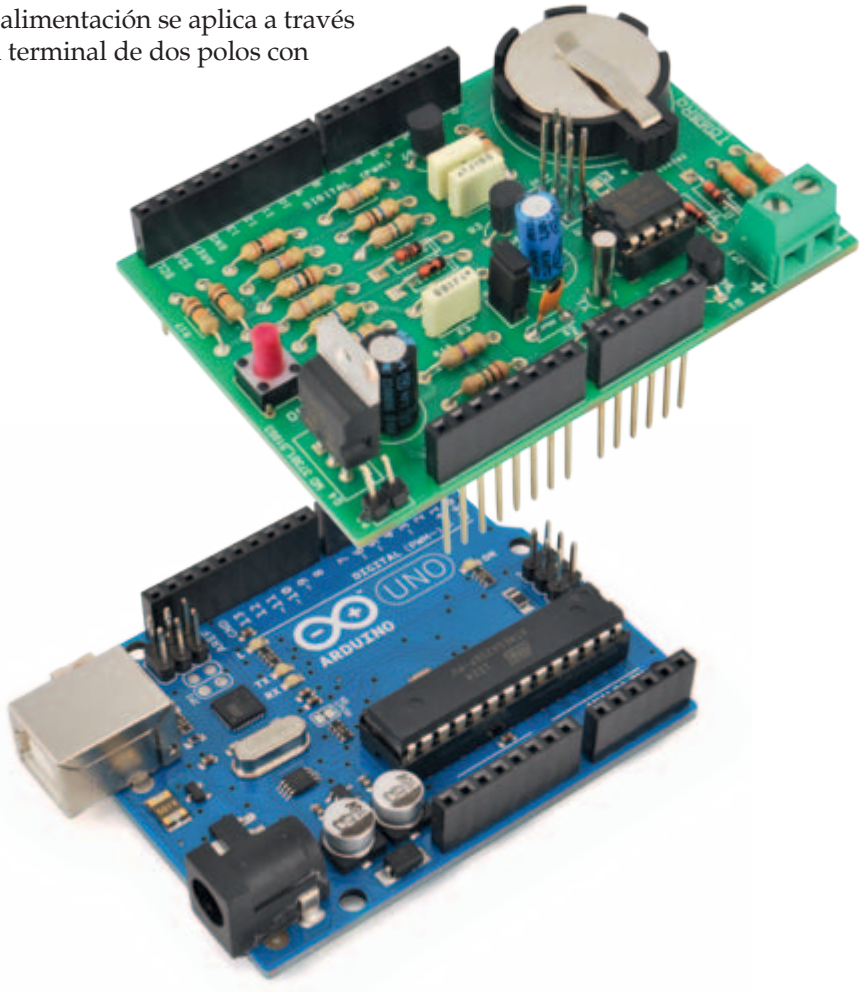
Fig. 1

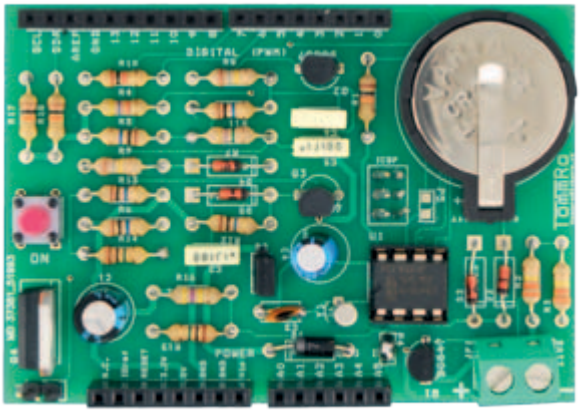
zación de las salidas, Arduino programa el reloj indicando el instante en cual tendrá que ser despertado; después decide apagarse;
- por todo el tiempo restante, el sistema queda apagado y todo el mecanismo se repite en el siguiente encendido.

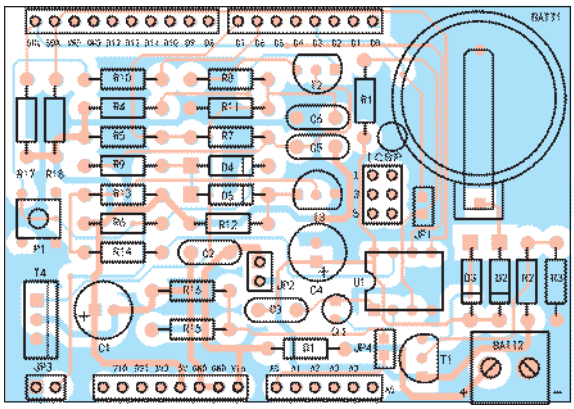
Es bastante intuitivo entender que si el software Arduino se realiza ad hoc, maximizando los tiempos de apagado, la autonomía de funcionamiento de un sistema Arduino alimentado por batería puede alargarse incluso algunos meses o años. Como con los acumuladores, hay que medirlo teniendo en cuenta el fenómeno de la autodescarga.
Claramente el sistema es más adecuado en condiciones en las cuales no es necesario un control en tiempo real, para el análisis de condiciones de peligro en el cual los tiempos deben ser mínimos o para sistemas que deben estar siempre online no parece muy adecuado; está sin embargo pensado para situaciones en las cuales se deben monitorizar medidas físicas que permanecen estables o varían lentamente en el tiempo (por ejemplo la temperatura de una habitación o el ambiente, los parámetros meteorológicos, la luminosidad ambiental, etc…).
Dado que el shield contiene en su interior el chip PCF8593T, puede utilizarse también como shield RTC para proporcionar a Arduino las funcionalidades típicas de un reloj y calendario.
El shield utiliza dos pines digitales (D6 y D7) para la gestión del sistema de encendido; el puerto I²C se utiliza para la comunicación/programación del componente PCF8593T. Hemos previsto la posibilidad (a través de un puente) de usar el pin analógico A0 para leer la tensión de alimentación proveniente de la batería, de tal manera que conozcamos el nivel de carga y eventualmente enviar informes del caso o apagar definitivamente el sistema o partes del mismo.
En el shield, además, está presente un pulsador para encender Arduino de manera forzada (manualmente, mientras se encuentra en un intervalo de apagado); claramente las condiciones serán previstas en fase de escritura del firmware. Tenemos también un jumper que si lo insertamos, permitirá mantener Arduino siempre encendido; eso puede ser útil en algunos casos particulares.

ESQUEMA ELÉCTRICO

Empezamos a analizar el esquema eléctrico del shield. La tensión de alimentación se aplica a través del terminal de dos polos con las siglas BATT2; el diodo D1 funciona como protección de la inversión de la polaridad de la alimentación.
El corazón del circuito es el integrado U1, un PCF8593T producido por NXP; se trata de un Real Time Clock programable, que para funcionar utiliza un cuarzo de 32.768 Hz (Q1), gracias al cual se marca el reloj del oscilador interno. La frecuencia de oscilación se establece con la ayuda del condensador C3, mientras C4 ayuda a nivelar la tensión de alimentación. El reloj/calendario puede alimentarse tanto por la batería principal del sistema, como por la pila de botón CR2032 presente en el shield; en el primer caso debe







Lista de materiales

R1: 10 kohm
R2: 330 kohm
R3: 330 kohm
R4: 4,7 kohm
R5: 10 Mohm
R6: 10 Mohm
R7: 4,7 Mohm
R8, R9: 180 kohm
R10: 1 kohm
R11: 10 Mohm
R12: 10 Mohm
R13: 10 Mohm
R14: 1 Mohm
R15: 1 Mohm 1%
R16: 470 kohm 1 %
R17, R18: 10 kohm
C1: 220 μF 16 VL electrolítico
C2, C5: 100 nF 100 VL poliéster
C3: 22 pF cerámico
C4: 100 μF 16 VL electrolítico
C6: 10 nF 63 VL poliéster
U1: PCF8593P
T1: BC547
T2: BC557
T3: BC557
T4: P36NF06
D1: 1N4007
D2: SD103
D3: SD103
D4: SD103
D5: SD103
P1: Microswitch
Q1: Cuarzo 32.768 kHz

Varios:
- Terminal 2 polos
- Tira de 2 pines macho (2 pz.)
- Jumper (2 pz.)
- Zócalo 2x4, 300 mils.
- Porta-batería para CR2032
- Tira de pines macho/hembra 2x3
- Tira de pines macho/hembra 6
- Tira de pines macho/hembra 8 (2 pz.)
- Tira de pines macho/hembra 10
- Batería CR2032
- Circuito impreso

cerrarse el jumper JP2 y la tensión llega desde la etapa basada en T1, que es un regulador lineal serie que asegura tensión constante independientemente (dentro de ciertos límites) del valor de la tensión de alimentación proporcionada en el terminal BATT2.
En el segundo, se abre el JP2 y cierra el JP4.
Si la pila de botón está insertada también cuando el circuito es alimentado por un acumulador externo, es útil para mantener alimentado el integrado RTC (U1) durante el cambio de la batería principal, de manera que no se pierda la configuración actual del reloj/calendario.
Los diodos D2 y D3 sirven para que una u otra fuente de alimentación hagan funcionar el Real Time Clock sin que una interfiera con la otra; esto es importante porque la tensión aplicada en BATT2 es por lo general superior a la de la pila de botón (3 V) y en ausencia de los diodos terminaría con la sobrecarga de la pila misma. Hay que señalar que ambos diodos son Schottky y esto no es una casualidad: los hemos elegido debido a su mínima caída de tensión (poco más de 0,2 voltios) de tal manera que se mantenga correctamente en funcionamiento U1 también con la pila de 3 voltios.
El PCF8593T dispone de un pin de salida INT, que es la verdadera peculiaridad, dado que, respecto a integrados como el más conocido DS1307 de Dallas, el componente de NXP dispone de una funcionalidad de despertador programable: a la hora y fecha indicada, el chip pone a nivel lógico alto la línea (pin 7) y con él es posible activar otras funciones. En nuestro caso, el pin sirve para alimentar Arduino, para encenderlo. Una vez alimentado, Arduino controla la propia línea

D7 de tal manera que hace entrar en estado ON el MOSFET T4 del shield, el cual conecta la masa de la batería con la masa de Arduino, manteniendo la tarjeta encendida. Cuando Arduino tiene que apagarse, deja ir D7 a nivel alto (T2 va en cortocircuito y deja bloqueado el MOSFET, el cual desconecta la masa de Arduino del negativo de la batería, abriendo el circuito de alimentación principal. Si Arduino tiene que permanecer encendido para continuar una tarea, mantiene la línea D7 a nivel bajo y entonces fuerza la saturación de T2, cuyo colector alimenta la puerta del MOSFET.
En nuestro esquema, la resistencia R5 sirve de pull-up cuando Arduino está apagado, de lo contrario la línea D7 estaría flotante (no garantizaría un estado lógico estable); si Arduino se apaga, R5 mantiene la base de T2 a nivel alto y la puerta de T4 a cero lógico.
El pulsador P1 permite encender de manera forzada Arduino: pulsándolo, se cierra a masa el nodo R6, R10, D4 y la base de T2 se pone a nivel bajo, lo que fuerza el PNP en saturación y el MOSFET en conducción; cuando recibe tensión, Arduino lleva a cero lógico D7. El mismo discurso vale para el jumper JP3: si se cierra, pone a masa el nodo D4, D5, R9, haciendo conducir T2 y T4.
Los diodos (son también Schottky, con el fin de minimizar la caída de tensión sobre ellos) D4 y D5 forman una puerta lógica AND que permite llevar a nivel bajo a la base de T2 tanto al pulsador P1, como al transistor T3 (controlado por la salida INT del PCF8593T) sin que los dos circuitos influyan en los demás. D4 sirve también para aislar la línea D6 de Arduino cuando el transistor T3 pasa a saturación.
Concluimos la descripción del shield analizando las líneas de los pines de conexión de Arduino, aún no descritas: se utilizan los contactos del puerto I²C (SDA e SCL, dotados de resistencias pull-up, que son respectivamente R17 e R18) que serán utilizadas para la comunicación con el PCF8593T; el contacto D6 sirve para decir a Arduino quién ha pedido el encendido, mientras la A0 se usa, si el jumper JP4 se cierra, para permitir a Arduino tomar el control de la tensión de la batería (el divisor R15/R16 baja la tensión de alimentación de manera que no supere la máxima permitida por las líneas analógicas).

LIBRERÍA ARDUINO BATTERY SHIELD

Como hemos visto en el artículo, para el correcto funcionamiento del sistema es necesario que todo el software sea escrito teniendo en cuenta cómo se realiza el mecanismo de encendido y apagado de la electrónica.
Para facilitar la integración por parte de los usuarios finales, hemos realizado una librería software que implementa las funcionalidades básicas. Se definen tanto las funciones de "bajo nivel" (entre ellas la inicialización del sistema, encendido, auto alimentación y apagado de Arduino, lectura del valor analógico de la tensión de alimentación) como aquellas de "nivel medio", entre las cuales está el arranque del chip PCF8593T y las relativas a la programación de los instantes de encendido. Se definen también las funciones de "alto nivel" típicas de un componente reloj/calendario, es decir, programación y lectura de fecha y hora, verificación del nuevo año (incremento del año), cambio

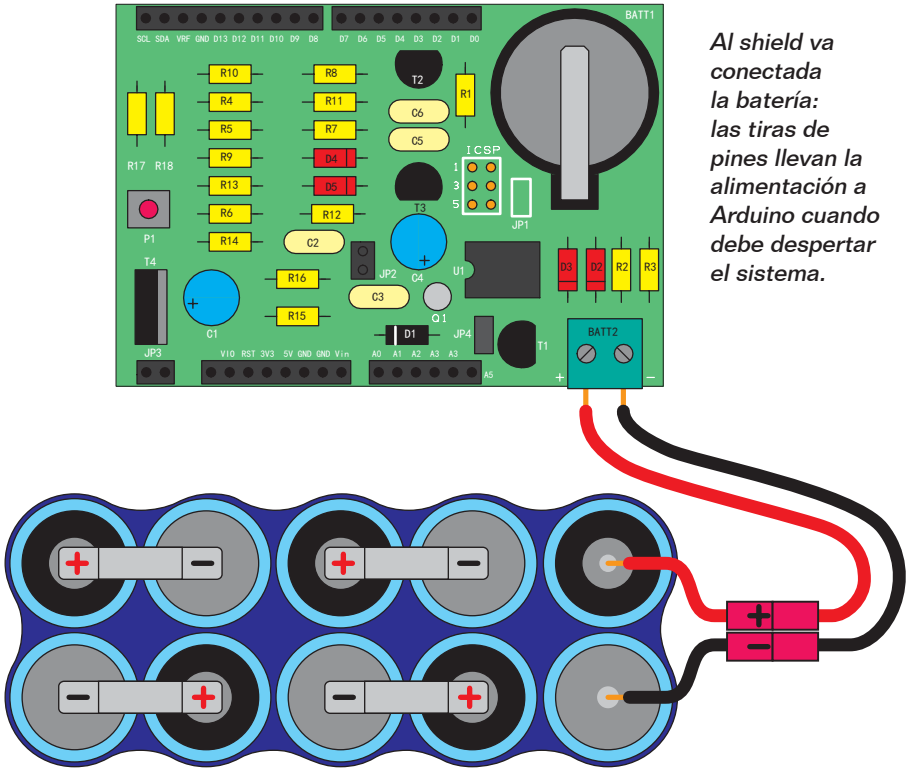
Funciones de la **librería**

Para la gestión de las funcionalidades del Battery shield es necesario una librería que emplea las funciones aquí descritas.

- **void begin()**: inicializa el hardware y software Arduino necesarios para el shield.
- **void powerON()**: configura el hardware Arduino para que se alimentado.
- **void turnOFF()**: configura el hardware Arduino para el apagado.
- **bool onFromButton()**: verifica si Arduino ha sido encendido a causa de presión del pulsador de encendido.
- **float getBatteryVoltage()**: lee el valor de la tensión de la batería de alimentación.
- **void startSecAlarmPCF8593, void startMinAlarmPCF8593, void startHourAlarmPCF8593**: seleccionamos el instante del próximo evento de encendido (es posible especificar por cuantos segundos, minutos o incluso horas Arduino debe permanecer apagado antes de volver a despertarse); como entrada se aceptan valores comprendidos entre 1 y 99 extremos incluidos.
- **void writeClockANDDataPCF8593()**: programa la fecha y hora actual (pasados como parámetros horas, minutos, segundos, día, mes y año) del RTC.
- **void readClockANDDataPCF8593()**: lee la fecha y hora actual indicada por el RTC.
- **bool checkProgrammedPCF8593()**: testea si el reloj/calendario de RTC está encendido y programado.
- **bool checkHourLegPCF8593():** verifica si por un determinado día debe ser activa la hora solar o legal.
- **void checkNewYear()**: testea si el calendario ha pasado el fin de año y actualiza el número de año.
- **byte getDayOfWeek()**: determina el día de la semana de una fecha.
- **bool YearBisestile()**: determina si un determinado año es bisiesto.

automático desde la hora legal a solar y viceversa, etc.
Como veremos más adelante, analizando el ejemplo de programación, utilizando estas funciones no debéis preocuparos de la gestión directa del hardware, ya que podéis contar con una interfaz software que se encarga y enmascara los detalles relativos.
La librería define un objeto "Battery" que tiene los siguientes métodos públicos (es decir, utilizables en el programa Arduino).

- **void begin()**: inicializa los pin D6 y D7 como entrada y salida digitales respectivamente y además enciende el reloj si este es detectado como parado.
- **void powerON()**:pin D7 a nivel bajo de manera que Arduino pueda permanecer encendido.
- **void turnOFF()**: al contrario de **powerON()**,D6 a nivel alto de manera que Arduino pueda apagarse.
- **bool onFromButton()**: en el análisis del esquema eléctrico hemos encontrado el pulsador P1, que sirve para encender Arduino pasando la programación del reloj; para distinguir si el encendido ha sido efectuado por el PCF8593T o por la presión del pulsador se utiliza el pin digital D6 de Arduino. La función **onFromButton()** testea el estado de D6 y por consiguiente identifica si el encendido deriva del pulsador (resultado de la función true) o por el reloj (resultado false).
- **float getBatteryVoltage()**: como hemos ya hemos, es posible utilizar el pin analógico A0 de Arduino para controlar la tensión de alimentación de la batería. Esta función se puede utilizar para leer tal valor; la función ejecuta ya todas las conversiones y tiene en cuenta las posibles caídas de tensión debidas a distintos componentes (en particular al diodo D15) y retorna en formato float (número en coma flotante) la tensión de la batería.
- **void startSecAlarmPCF8593**, **void startMinAlarmPCF8593**, **void startHourAlarmPCF8593**: con estas funciones se define el instante del próximo evento de encendido. En particular, se puede especificar (tomando como referencia el instante de ejecución de las funciones) con respecto a segundos, minutos u horas, cuánto debe permanecer apagado Arduino antes de ser despertado por el reloj. Las funciones aceptan como entrada un parámetro de un byte cuyos valores válidos son solo aquellos comprendidos entre 1 y 99, ambos incluidos.
- **void writeClockANDDataPCF8593()**: como el integrado PCF8593T, además de ser utilizado para determinar los instantes de encendido, puede ser utilizado genéricamente como RTC y reloj/calendario, esta función permite programar la fecha y hora actual (pasados como parámetros horas, minutos, segundos, día, mes y año). La función determina y programa en automático el día de la semana y si está activa la hora solar o la legal.
- **void readClockANDDataPCF8593()**: al contrario de la anterior función, permite leer la fecha y hora actual indicada por el chip PCF8593T. Como calendario retorna el día, mes y año, más el día de la semana; como reloj retorna la hora, los minutos, los segundos y las centésimas de segundo.
- **bool checkProgrammedPCF8593()**: permite testear si el reloj/calendario del chip PCF8593T han sido encendidos y programados. En caso afirmativo retorna true; en



Al shield va conectada la batería: las tiras de pines llevan la alimentación a Arduino cuando debe despertar el sistema.

caso negativo retorna false (en este caso encenderá llamando a la oportuna función).

- **bool checkHourLegPCF8593()**: permite verificar si para un determinado día (identificado por día de la semana, día y mes) debe estar activa la hora solar o legal (útil para ejecutar en automático el cambio de hora).
- **void checkNewYear()**: testea si el calendario ha pasado del final del año y en caso afirmativo, actualiza (incrementa de 1) el número del año.
- **byte getDayOfWeek()**: determina el día de la semana de una fecha (identificado por día, mes y año).
- **bool YearBisestile()**: determina si un determinado año es bisiesto.

PRIMER EJEMPLO FIRMWARE

Para entender el funcionamiento y la lógica de programación a utilizar con el battery shield, veamos dos ejemplos de código Arduino, el primero de los cuales está escrito en el **Listado 1**. Este, simplemente programa el battery shield para que Arduino se mantenga casi siempre apagado, encendiéndose cada 10 segundos y simula la ejecución de un programa (a través delay), después se apaga de nuevo en espera del próximo ciclo de ejecución.
Después de incluir las librerías utilizadas (en particular la "Battery.h" especifica del shield) se define el pin digital de gestión del LED presente en la tarjeta Arduino y además se define un objeto Battery que se utilizará para la gestión del mismo shield.
En la función de setup de Arduino (que, recordamos, es la primera que se ejecuta al encender) se programa el pin del LED y este se enciende; a continuación se arranca la librería Wire y la Serial (esta última utilizada para la depuración y seguir mejor el flujo del software).
Después se inicializa el objeto "batteryShield" a través de la función **batteryShield.begin()**; esta última ha sido escrita de manera que inicialice todos los recursos (tanto hardware como software) necesarios para gestionar el shield (de manera que "esconda" al usuario todos los detalles implementados de bajo nivel).
La próxima instrucción es la "batteryShield.powerON()" que configura el shield (en particular

Listado 1

```
//*****************************************************************
//* Inclusión librerías   *
//*****************************************************************
#include <Battery.h>
#include <Wire.h>
#include <Serial.h>

//*****************************************************************
//* Definición pin I/O Arduino     *
//*****************************************************************/
// Led tarjeta Arduino
const int pinBoardLed = 13;

//*****************************************************************
//* Variables programa    *
//*****************************************************************/
// Objeto battery shield
Battery batteryShield;

//*****************************************************************
//* Código programa       *
//*****************************************************************/

// Inicialización tarjeta
void setup() {
  // Inicializo I/O
  pinMode(pinBoardLed, OUTPUT);
  // Enciendo led Arduino
  digitalWrite(pinBoardLed, HIGH);

  // Arranque librería Wire
  Wire.begin();
  // Arranque puerta serie
  Serial.begin(115200);

  // Inicializo battery shield
  batteryShield.begin();
  // Autoalimentación Arduino
  batteryShield.powerON();

  // Si hay encendido de pulsador
  if (batteryShield.onFromButton() == true)
    Serial.println("Acceso da pulsante");
  // Si hay encendido no de pulsador
  else
    Serial.println("Encendido de reloj");

  // Observo/señalo tensión batería
  Serial.println(batteryShield.getBatteryVoltage());
}    // Cerrado función setup

// Programa Principal
void loop() {
  // Attesa (tiempo tarjeta encendida)
  delay(2500);

  // Programo reloja para próximo encendido (en 10 segundos)
  batteryShield.startSecAlarmPCF8593(10);
  // Apago led Arduino
  digitalWrite(pinBoardLed, LOW);
  // Apago Arduino
  batteryShield.turnOFF();
}    // Cierre función loop
```

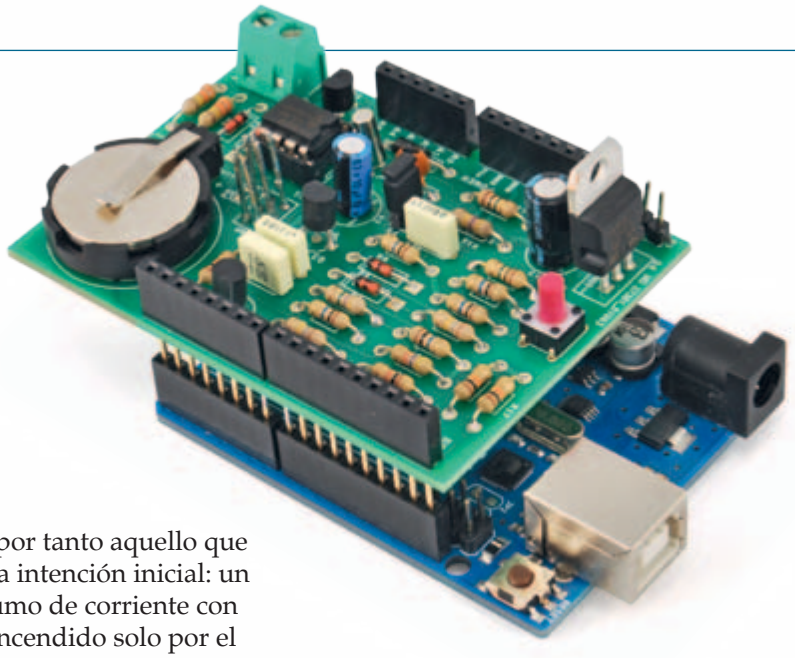
Listado 2

```
//*************************************************************************
//* Inclusión librerías   *
//*************************************************************************/
#include <EEPROM.h>
#include <Battery.h>
#include <Wire.h>
#include <Serial.h>

//*************************************************************************
//* Definición pin I/O Arduino     *
//*************************************************************************/
// Led tarjeta Arduino
const int pinBoardLed = 13;

//*************************************************************************
//* Variables programa    *
//*************************************************************************/
// Objeto battery shield
Battery batteryShield;

//*************************************************************************
//* Código programa       *
//*************************************************************************/
// Inicialización Tarjeta
void setup() {
  // Inicializo I/O
  pinMode(pinBoardLed, OUTPUT);
  // Enciendo led Arduino
  digitalWrite(pinBoardLed, HIGH);

  // Arranque librería Wire
  Wire.begin();
  // Arranque puerta serie
  Serial.begin(115200);

  // Inicializo battery shield
  batteryShield.begin();
  // Autoalimentación Arduino
  batteryShield.powerON();
}    // Cierre función setup

// Programa Principal
void loop() {
  int contatore;

  // Leo contador de EEPROM
    contatore  =  (((int)  (EEPROM.read(0x0000))  <<  8)  +  (int)  (EEPROM.
read(0x0001)));
  // Incremento contador
  contatore++;
  // Envío contador
  Serial.println(contatore);
  // Memorizo contador
  EEPROM.write(0x0000, (byte) (contatore >> 8));
  EEPROM.write(0x0001, (byte) (contatore & 0x00FF));

  // Programo reloj para próximo encendido (en 30 segundos)
  batteryShield.startSecAlarmPCF8593(30);
  // Apago led Arduino
  digitalWrite(pinBoardLed, LOW);
  // Apago Arduino
  batteryShield.turnOFF();
}    // Cierre función loop
```

el pin digital D7) de manera que Arduino es alimentado constantemente (en la práctica es una especie de autoalimentación en la cual Arduino decide si permanecer encendido o no). Llamando "batteryShield.onFromButton()" se identifica la causa del encendido (pulsador de ON o encendido programado por reloj) y envía un mensaje de debug por el puerto serie. Después se lee y envía por el puerto serie la tensión de alimentación de la batería; el valor visualizado es ya el real de la batería, teniendo en cuenta todas las caídas de tensión debidas a diodos y varios divisores resistivos. A este punto, el software Arduino está en ejecución y la tarjeta, en estas condiciones, se mantiene siempre alimentada (la batería lo permite). Terminada la función "setup()" se ejecuta el bucle "loop()"; la primera instrucción es un delay (de alrededor 2,5 segundos) que simula el funcionamiento de la rutina main de un programa genérico (lectura de entradas, ajuste de salidas, etc.). A continuación se ha programado que el próximo instante de encendido (instrucción "batteryShield.startSecAlarmPCF8593(10)") será en 10 segundos, entonces se apaga el LED de Arduino y después se quita la alimentación de la tarjeta en su conjunto (instrucción "batteryShield.turnOFF()"). Esta última instrucción es bloqueadora, es decir, además de programar el pin digital D7, entra en un ciclo infinito del cual no saldrá más (hasta que físicamente no se quite la alimentación, cuando la tarjeta se apaga). Transcurrido el tiempo programado de 10 segundos, el PCF8593T vuelve a encender la tarjeta y todo el mecanismo se repite (el software reinicia desde el setup como si fuese el primer encendido).

SEGUNDO EJEMPLO FIRMWARE

Veamos ahora un segundo ejemplo de código Arduino (**Listado 2**) que utiliza las funcionalidades del shield para obtener un encendido cada 30 segundos y memoriza (en EEPROM) un contador (a 16 bit) de encendido. También en este caso se incluyen librerías (en particular la "Battery.h" específica del shield); se definen el pin digital correspondiente al LED de Arduino y después un objeto Battery, que se utilizará para acceder al shield.



En la función de setup de Arduino se programa y se enciende el pin del LED; después se arranca la librería Wire y la Serial (esta última utilizada como debug para aclarar mejor el flujo del software). A continuación se inicializa el objeto "batteryShield" (función batteryShield.begin()) de manera que se inicializan todos los recursos (tanto hardware como software) necesarios para la gestión del shield. La próxima instrucción es la "batteryShield.powerON()" que indica al shield de mantener Arduino constantemente alimentado.
En este punto el software Arduino está en ejecución y la tarjeta, en estas condiciones, se mantiene alimentadade forma continua. Terminada la función "setup()" se ejecuta la "loop()"; las primeras instrucciones leen los primeros dos byte de la EEPROM y los utilizan para "construir" el contador de 16 bit de los encendidos; tal contador se va incrementando y actualizando el valor en la EEPROM (el mismo contador es enviado sobre el puerto serie para la depuración). A continuación se programa que el próximo instante de encendido (instrucción "batteryShield.startSecAlarmPCF8593(30)") será en 30 segundos, se apaga el led de Arduino y después se quita la alimentación de la tarjeta en su conjunto (instrucción "batteryShield.turnOFF()"). Pasado el tiempo programado de 30 segundos, el PCF8593T vuelve a encender la tarjeta y todo el mecanismo se repite (el software vuelve a empezar con el setup como si fuese su primer encendido). El tiempo de ejecución de este ejemplo es seguramente más rápido que el anterior; la tarjeta quedará encendida pocos instantes mientras la mayor parte del tiempo estará apagada. Hemos realizado por tanto aquello que era nuestra intención inicial: un bajo consumo de corriente con Arduino encendido solo por el tiempo necesario.

REALIZACIÓN PRÁCTICA

El hardware del sistema presentado está constituido por una tarjeta Arduino (suponemos la Uno Rev. 3) y por un Battery shield; este último suministrado como kit de montaje que puedes encontrar en nuestra web (*www.nuevaelectronica.com*) y todos sus componentes son de montaje convencional, lo que permite la realización esté al alcance de todos.
Adquirido lo necesario, se puede empezar por los componentes de perfil más bajo (resistencias, diodos, condensadores cerámicos y electrolíticos, puentes, etc..) y después pasar al soporte de la pila, los terminales de 2 polos. Proceder entonces con el integrado PCF8593P (primero soldar el zócalo, después insertar el integrado prestando atención a la posición de la muesca que indica el pin 1) y finalmente las tiras macho/hembra para la inserción del shield en Arduino. Completado el montaje, colocar la pila de botón en su lugar.

CONCLUSIONES

El shield presentado permite realizar sistemas Arduino alimentados por batería que, si se programan correctamente, pueden funcionar sin sustituir la batería durante bastantes años.
El shield ha sido proyectado para Arduino Uno Rev. 3, el nivel de iniciación de Arduino, pero se puede adaptar a las otras placas Arduino. Los sketch de ejemplo presentados en estas páginas tienen el objetivo de explicar cómo utilizar las librerías desarrolladas; no representan aplicaciones reales pero pueden ser utilizados como punto de partida para vuestros proyectos.

(186075)

el MATERIAL

Todos los componentes utilizados para realizar el shield descrito en este artículo son fáciles de encontrar en el mercado. En la web de Nueva Electrónica se puede adquirir el kit de este proyecto (cod. BATTERYSHIELD, 19,50 Euros) y también la tarjeta Arduino Uno Rev.3 (24,50 Euros). El sketch para realizar el proyecto también se puede descargar gratuitamente de la misma web.

Precios IVA incluido sin gastos de envío.
Puede hacer su pedido en:
www.nuevaelectronica.com
pedidos@nuevaelectronica.com

Acompáñanos en esta nueva etapa

¡Suscríbete!

Elige tu formato: Impreso, Digital o Ambos

Nueva Electrónica se publica 12 veces al año y puedes adquirirla directamente en nuestro kiosko web o suscribirte por un periodo de uno o dos años consiguiendo un considerable descuento respecto al precio de cubierta.

Visítanos ahora en www.nuevaelectronica.com

Descuentos especiales para colectivos, estudiantes, centros de enseñanza y bibliotecas.

ATENCIÓN: Si ya eras suscriptor de Nueva Electrónica y vas a renovarla ahora, tienes que hacerlo directamente en la sección Suscripciones de nuestra web siguiendo las instrucciones de pago que allí se indican. Si tienes alguna duda, escríbenos a revista@nuevaelectronica.com

ARDUINO YÚN: LA VIDA ES MÁS FACIL

MARCO MAGAGNIN

Aquí está la penúltima tarjeta de Arduino. El ADN es el de la familia, pero la evolución la ha abierto a las nuevas exigencias: Linux embebido, conectividad ethernet y WiFi. Para probarla, hemos desarrollado una aplicación que envía Tweets al producirse eventos locales.

Aunque YUN ya lleva algún tiempo en el mercado, somos muchos los que no habíamos tenido oportunidad de cacharrear con él, así que hemos decidido compartir con vosotros nuestra primera experiencia con esta placa. Como anunciaron, la tarjeta integra las características de un Arduino Leonardo estándar, con una sección hardware adjunta capaz de albergar una distribución GNU/Linux completa, en este caso OpenWRT, específicamente destinada a la conectividad con Internet (es la distribución open source utilizada en los router y access point WiFi). Con el producto en la mano, lo primero que nos preguntamos es: ¿seguro que me habéis dado una Yún? La caja tiene las mismas dimensiones que aquella

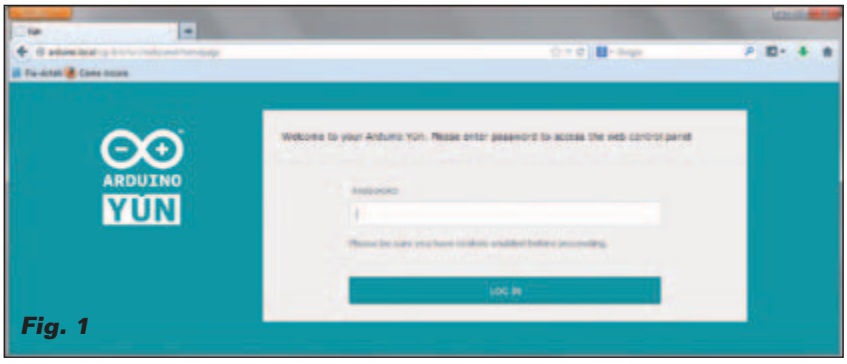


Fig. 1

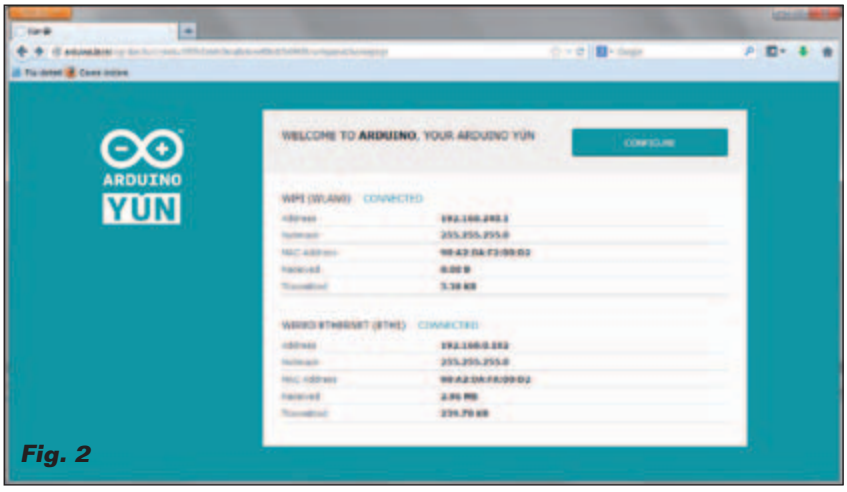


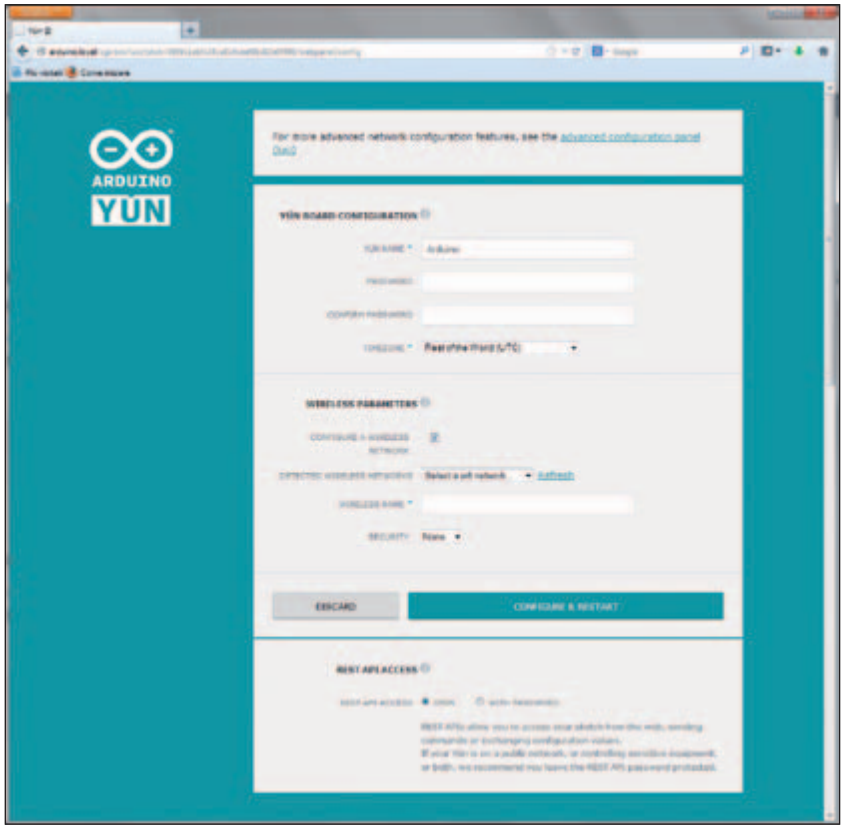
Fig. 2

de Arduino Uno. Abierta la caja nos hemos asegurado: sí, es una Yún, aunque tenga las mismas dimensiones de una Arduino Uno (obvio, porque tiene que ser compatible con los mismos shield). Teniéndola entre las manos se ven dos entradas USB, una Ethernet, una entrada para micro SD-Card y los conectores estándar de un Arduino. Además están presentes tres pulsadores y un conector para una antena externa a la sección WiFi. Todo rodeado por una buena cantidad de electrónica SMD situada en ambas caras del circuito impreso.
Queriendo tener un enfoque típico de experimentadores provenientes del mundo GNU/Linux, antes de profundizar más en las características de la tarjeta probemos a ver qué sucede, en todo caso, ¡después leeremos el manual!
Entonces, conectamos un cable de red a la toma Ethernet de Arduino Yún y lo alimentamos mediante un cargador a 5V con toma micro USB (para entendernos, la misma que se usas para recargar los teléfonos) al conector micro USB en el centro de la tarjeta. El LED ON se enciende en verde y poco después el LED WAN cambia a amarillo. ¿Qué quiere decir? Abrimos un buscador y escribimos la dirección:

http://arduino.local

(en realidad es necesario leer un poco en el "getting started" que hay en la página web *www.arduino.cc* dedicada a Yún). Sorpresa: obtenemos la página que se en la **Fig. 1**. Ponemos la contraseña "arduino" (es la contraseña del usuario por defecto ("root") y vemos que sucede. Obtenemos la página que vemos en la **Fig. 2,** que nos muestra la configuración de las interfaces de red activas. Esta página ya nos dice muchas cosas. Yún nace con un sistema operativo GNU/Linux

Fig. 3



instalado, con los interfaces de red preconfigurados y un servidor web activo con la aplicación LuCI ya instalada para configurar el funcionamiento en GNU/Linux.
A propósito: ¿habéis notado que en el primer arranque hemos usado un nombre (*arduino.local*) y no una dirección IP?
Esto significa que no es necesario averiguar la dirección IP asignada a Yún por el servidor DHCP de nuestra red, y si podemos acceder directamente con el nombre host de la tarjeta, debido a que la distribución GNU/Linux de Yún, orientada a la conectividad, está dotada de un servidor DNS interno.
En la **Fig. 2** luego veremos cuál es la dirección IP asignada a la tarjeta WAN y la configuración de la sección WiFi, ya activada y preconfigurada como punto de acceso con una IP estática. Esto significa que podemos conectarnos a la Yún

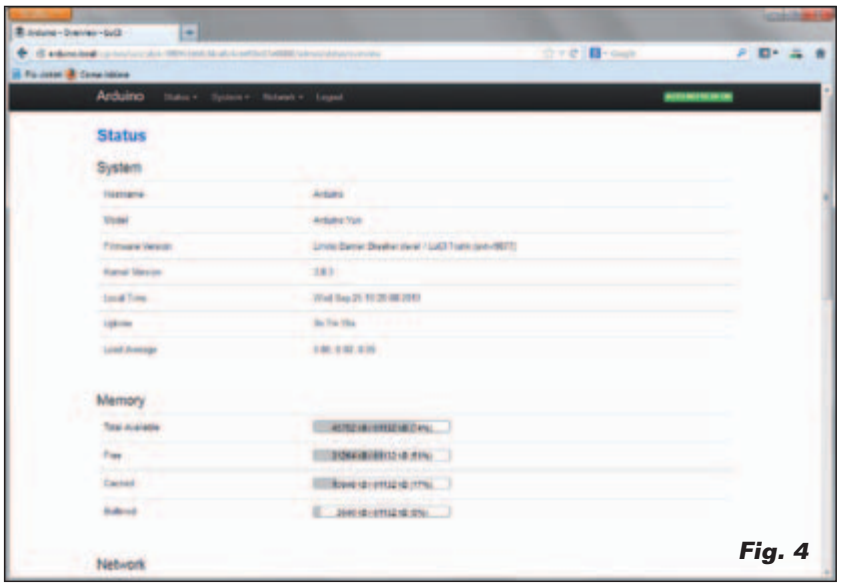


Fig. 4

Continuamos curioseando y hacemos clic en el pulsador grande "Configure" arriba a la derecha de la página: obtenemos otra página de configuración (**Fig. 3**) que muestra, arriba a la derecha, un link que atrae nuestra atención.
Pero primero aprovechamos para configurar la zona horaria exacta y seleccionamos el check box

secciones de administración del sistema GNU/Linux (**Fig. 5**).
A propósito, la distribución personalizada para Arduino Yún se llama "Linino" y, como ya hemos dicho y cómo podemos ver en el menú de configuración, ha derivado de Open WRT. Si por curiosidad vamos a la sección "Network" y "Wifi", en el inte-


Fig. 5

también por WiFi, simplemente conectándonos a la red con SSID "Arduino Yun-<mac-address-dela-tarjeta>". Sin monitor, sin teclado ni ratón, ninguna conexión USB con el PC y, si queremos, ningún cable de red. Todo esto con el objetivo de facilitar las cosas para aquellos que son principiantes.

"Open" en la sección "Rest API Access"; luego hablaremos sobre esto.
Bien, pinchamos sobre el link "advanced configuration panel" y entramos en un verdadero y propio ambiente de administración, se puede ver en la **Fig. 4**. Desde el menú arriba, al estilo de Google, podemos acceder a las diferentes

rior de la página, presionamos el pulsador "Edit" y desplazamos la página un poco hacia abajo, encontramos la sección que nos permite configurar el módulo WiFi de la tarjeta en las diferentes modalidades (Access Point, Client, Ad Hoc, ecc). Las otras secciones nos permiten aplicar reglas de

accesos de seguridad (Firewall), reglas de Routing, DHCP y DNS. En definitiva, nos encontramos frente a un verdadero y propio dispositivo de conectividad que nos permitirá resolver una vasta gama de aplicaciones, de los requisitos de procesamiento y de configuración de red.

Una última curiosidad: probamos a conectarnos a la tarjeta utilizando los clientes SSH Putty y WinSCP. Nos conectamos con la dirección asignada por el DHCP de nuestra red (lo vemos en la página de la **Fig. 2**), con usuario "root" y contraseña "arduino". Obtenemos los resultados que vemos en la **Fig. 6** y **Fig. 7**, que nos demuestran que también el servidor SSH está activo y funcionando "out of the box".

Con lo que hemos visto ya, seria material suficiente para escribir un libro sobre las posibilidades de esta tarjeta, que contiene una verdadera y propia arquitectura integrada que reúne en pocos centímetros cuadrados un punto de acceso WiFi, un sistema GNU/Linux con aplicaciones web ya disponibles y un microcontrolador Arduino capaz de interactuar con el mundo exterior de las maneras que ya conocemos. Casi nos olvidamos de que se trata de una tarjeta Arduino...

En este momento no podemos dejar de subrayar el valor didáctico de este dispositivo, que, al enfoque simplificado de la programación de los microcontroladores, añade la posibilidad de acercarse de manera muy simplificada al mundo GNU/Linux en su configuración embebida y al mundo de la conectividad inalámbrica y de red en general, en sus diferentes configuraciones. Profundicemos estos argumentos, aunque no son de corte específicamente electrónico como en artículos específicos; somos conscientes que el mundo de la electrónica, de la informática y de las arquitecturas de red se están fundiendo en un todo fuertemente integrado.

Desde este punto de vista, Arduino Yún representa el punto de partida óptimo para empezar a experimentar la integración de las distintas disciplinas electrónicas, informáticas y de red, además de para realizar aplicaciones que unen la adquisición y elaboración de datos provenientes de sensores con la publicación y gestión de los mismos por el mundo exterior. En estos momentos, la máxima expresión de esta tendencia es el dispositivo myRIO de National Instruments, que integra en un único chip la potencia en tiempo real de un FPGA y un procesador que aloja un GNU/Linux con la conectividad entre los dos, basada en 16 canales DMA ultra veloces. Es ahora el momento de pararnos y profundizar las características generales de la Yún. Tomándola de nuevo entre las manos notamos

La necesidad de la conectividad de red cableada e inalámbrica para las tarjetas embebidas se siente no solo por el equipo Arduino, sino también por cualquiera que desarrolle prototipos de de sistemas embebidos. Veamos que se mueve en este sector.

UDOO

El mismo día del anuncio por parte del equipo de Arduino de la disponibilidad de la Yún, sobre el blog de Udoo se anunciaba el inicio de la producción de la tan deseada placa Udoo, un proyecto financiado gracias a Kickstarter, con la intención de superar la pobre propensión de un sistema de microcontrolador como Arduino a la

Fig. 6

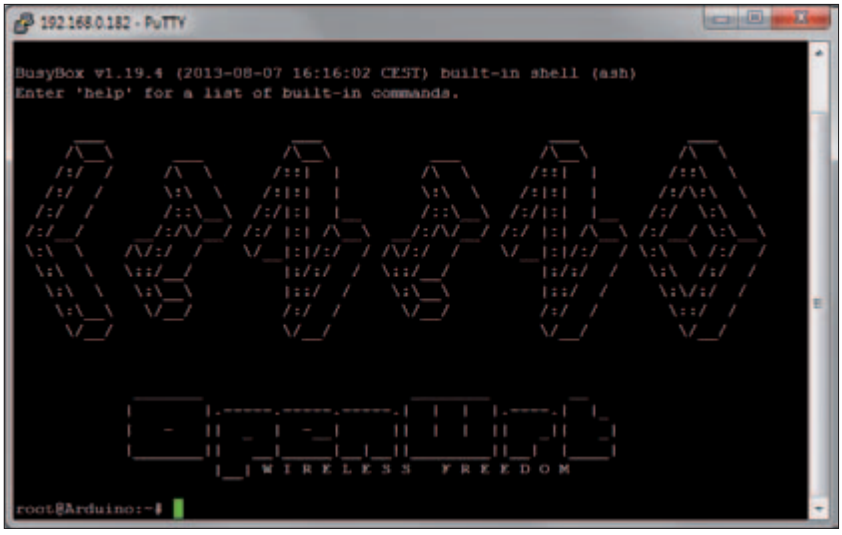
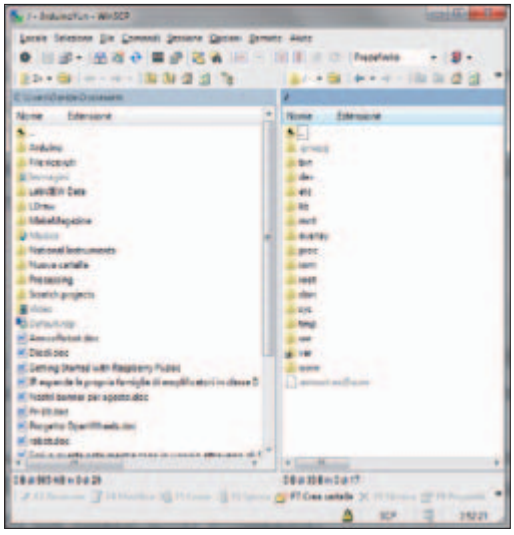


Fig. 7



conectividad (cosa usual para sistemas basados en Linux, que pueden proporcionar el acceso a Internet de manera natural). De hecho la tarjeta se acompaña de dos procesadores: un Freescale i.MX6 ARM quad o dual core (depende de la versión que elijáis) con 1 GB di RAM, para la sección GNU/Linux, y un Atmel SAM3X8E ARM Cortex-M3 para la sección de Arduino (el mismo que Arduino Due). La comunicación entre las dos secciones debería hacerse a través de un enlace serie dedicado. La tarjeta se acompaña de una versión personalizada de Ubuntu, pero gracias al apropiado slot para la tarjeta SD es posible instalar otra distribución Linux o Android. La tarjeta está dotada de módulo WiFi y expone también un conector Ethernet, una única salida HDMI, un puerto SATA, un controlador USB, un jack para los altavoces y uno para el micrófono.

NI myRIO

Un sistema embebido compacto y dotado de gran conectividad es myRIO de National Instruments, creado para ofrecer a los estudiantes la posibilidad de desarrollar rápidamente sistemas de ingeniería reales y completos. Basado en la misma tecnología de la plataforma NI CompactRIO, NI myRIO utiliza un SoC Xilinx Zynq-7020 que combina un procesador dual-core ARM ®Cortex™-A9 y un FPGA 7-Series de Xilinx con 28000 celdas lógicas programables. myRIO se programa en LabVIEW, con lo cual es rápido e intuitivo. NI myRIO dispone de 10 entradas analógicas, seis salidas analógicas, canales I/O audio y 40 líneas de I/O digitales. Además está dotado de WiFi en la tarjeta, de acelerómetro a tres ejes y diferentes LED programables.
La introducción de NI myRIO en la arquitectura LabVIEW RIO (Reconfigurable I/O) permite a National Instruments proporcionar instrumentos a todos los niveles de experiencia, desde el aprendizaje base de los estudiantes, hasta las soluciones más avanzadas para los diseñadores. Ideal para lecciones en clase y en laboratorio, NI myRIO incluye recursos didácticos gratuitos, es compatible con todos los NI miniSystems y con una vasta gama de actuadores y sensores de terceras partes. NI myRIO es programable en LabVIEW, C, C++ y otros entornos de desarrollo, de manera que es fácil de integrar en cursos didácticos sobre controles, la robótica, la mecatrónica y los sistemas embebidos.

otra característica muy útil: sobre los lados exteriores de los conectores para los shield están serigrafiadas las referencias principales de cada pin, que permiten utilizar la tarjeta sin necesitar el esquema de las conexiones. En la **Fig. 8** se puede ver la imagen de la tarjeta con las conexiones externas y los pulsadores de reset.
Pero procedamos en orden: del lado "microcontrolador", Yún utiliza el chip Atmega32U4, el mismo de Arduino Leonardo, del cual reproduce las características (16 MHz de reloj y 32 kB de RAM, de los cuales bootloader ocupa cuatro). En cuanto a la alimentación, no hay regulador a 5V, por lo que es necesario prestar atención de no alimentar la tarjeta con una tensión superior (puede causar daños irreparables), si se tiene la intención suministrar la alimentación a través del contacto Vin. En referencia a esto, la toma de alimentación en formato micro-USB es una ayuda: por su naturaleza, las alimentaciones de este tipo de salida suministran 5V. Esta micro-USB sirve también para conectar la tarjeta al PC para la programación tradicional por parte de Arduino. Pero es también posible ejecutar la carga de los sketch a través de la red, física o WiFi directamente desde el IDE.
A propósito: el IDE a utilizar, es descargable libremente (como siempre) desde la dirección web *http://arduino.cc/en/Main/Software*, es la versión *1.5.4 r2* ó superior, que resuelve un bug de funcionamiento bajo Windows 8. Existen también versiones para GNU/Linux y Mac OS X.
Otra diferencia de comportamiento respecto a Leonardo, está en el hecho que el microcontrolador no se resetea cuando se abre el monitor serie del IDE. Además

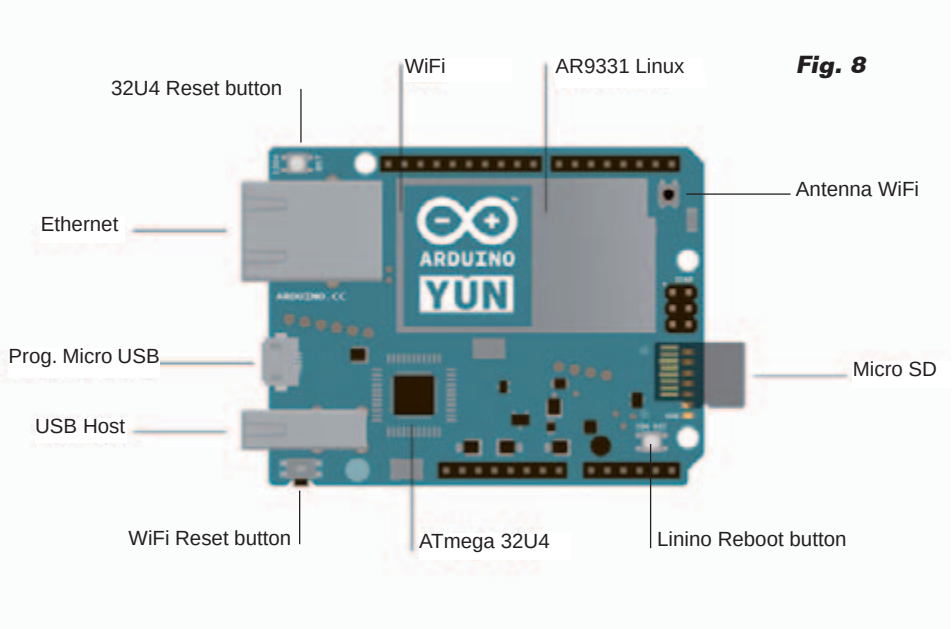
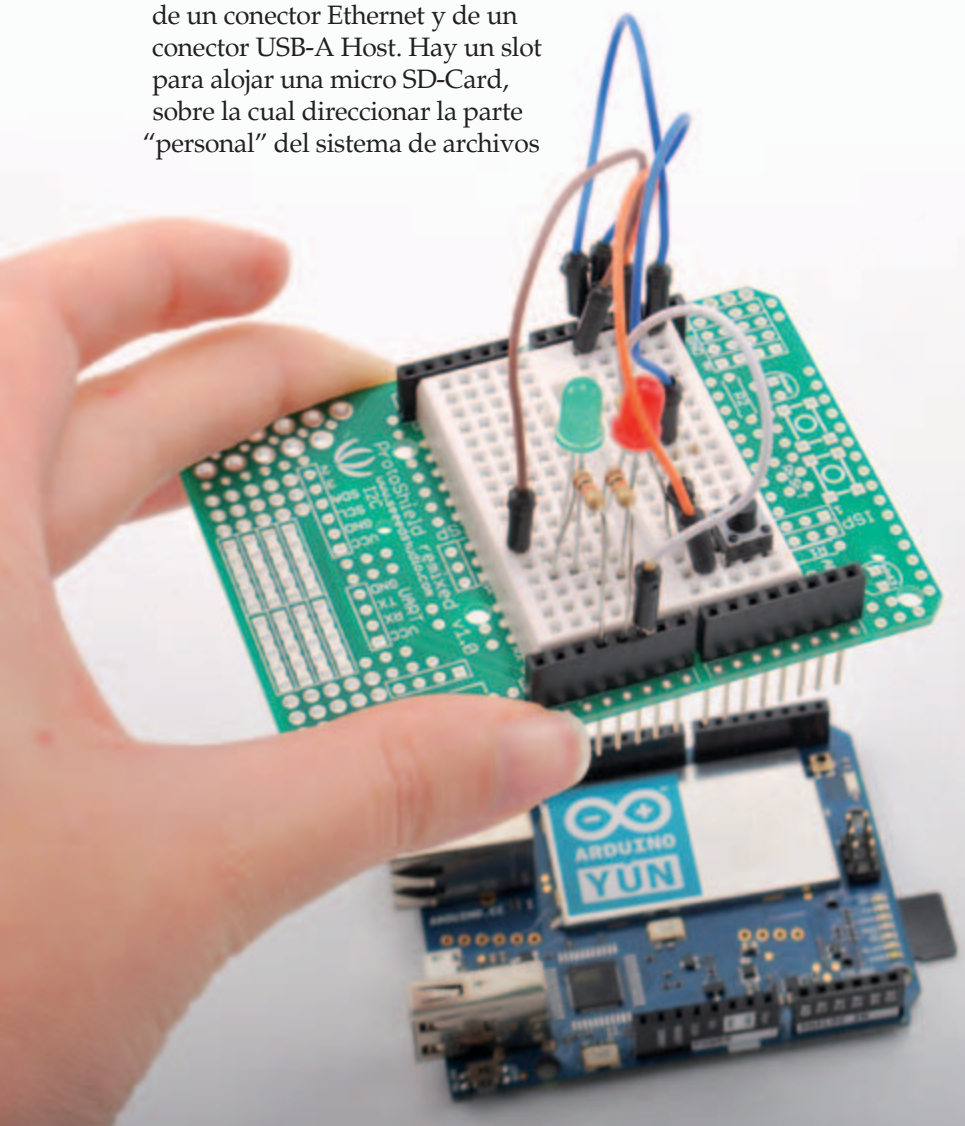


Fig. 8

no es posible utilizar la interfaz Serial1, reservada a la comunicación con la sección GNU/Linux. Para resetear el microcontrolador está el pulsador específico "32U4 Reset Button" al lado del conector Ethernet (ver **Fig. 8**).
La sección GNU/Linux se gestiona por el segundo procesador, un Atheros AR9331 a 400 MHz. Probablemente en la elección ha sido determinante la particular predisposición del SoC para albergar la distribución GNU/Linux OpenWRT, de la cual se ha derivado Linino, y por los consumos particularmente reducidos. La sección GNU/Linux está dotada de un conector Ethernet y de un conector USB-A Host. Hay un slot para alojar una micro SD-Card, sobre la cual direccionar la parte "personal" del sistema de archivos de la sección GNU/Linux, de manera que se ahorre la poca RAM NAND asignada al procesador Atheros (por otra parte caracterizada por un número de posibles reescrituras bastante limitado: cerca de 100.000). Para hacer esto es suficiente insertar una micro SD en el slot y crear dos carpetas que se reconocen automáticamente y se utilizan por el sistema operativo:

/arduino e
/data

En realidad, en el momento del boot el sistema operativo monta automáticamente todos los dispositivos de almacenamiento masivo externos denominados "sda", más alguna otra cosa. Veremos en profundidad el detalle el uso de estas carpetas.
En la sección GNU/Linux está presente también un módulo WiFi, conectado también a un procesador Atheros AR9331, que proporciona accesibilidad inalámbrica al sistema completo, con la posibilidad adicional de conectar una antena externa.
Vayamos ahora a los pulsadores de reset: el primero lo hemos visto ya y permite el clásico reset del microcontrolador Atmega32U4. Los otros dos se encargan del reset del procesador Atheros AR9331 y del módulo WiFi. La presión del pulsador "Linino reset button" permite ejecutar el reboot del sistema operativo Linino. Claramente todos los datos en memoria se perderán y todos los procesos se terminarán. El pulsador "WiFi reset button" tiene una doble función: pulsado brevemente resetea el módulo WiFi a la configuración de fábrica, ósea al funcionamiento como Access Point con IP estática igual a 192.168.240.1. Sin embargo, pulsando por un tiempo (5 segundos) el pulsador, se resetea el módulo WiFi. El reset del módulo WiFi hace también el reboot de Linino. Es importante darse cuenta que no debe presionarse la tecla más de 5 segundos, porque en cuanto se presiona más de 30 segundos, provoca la restauración del sistema operativo Linino con la versión original, o lo que es lo mismo la configuración de fábrica. Se ha elegido este funcionamiento para ayudarnos en el caso en que queramos recuperar el sistema después de algunas "torpes operaciones". La función de restauración nos da la libertad de experimentar sin temor de encontrarnos en situaciones sin salida: más



simple que así...
Restaurando el sistema claramente se pierden todas la configuraciones hechas después del primer encendido y otras actualizaciones ejecutadas sobre Linino.
Ahora digamos algunas palabras sobre los LED de la tarjeta. Arduino Yún dispone de los siguientes LED:

- USB - azul, indica actividad sobre el conector USB;
- WLAN (WiFi); WiFi activo;
- POWER; indica la presencia de alimentación;
- WAN; indica la presencia de una conexión de red Ethernet activa;
- Pin 13; reporta el estado del pin 13 (con tarjeta nueva parpadea por el programa Blink cargado en origen);
- Serial TX; señala actividad en el puerto serie en transmisión;
- Serial RX señala actividad en en puerto serie en recepción.

En la **Fig. 9** puedes ver el esquema de la arquitectura general de la tarjeta. Son claramente visibles las dos secciones "Arduino" y "Linino", con los periféricos asignados a cada uno y el "puente" de conexión indicado como "Bridge". Esta característica es la que permite a los dos ambientes trabajar conjuntamente y constituye una innovadora solución Linux embebido capaz de trasladar al mundo exterior todas las funcionalidades del microcontrolador Arduino, hasta ahora confinadas solo al ámbito local. A decir verdad, en el pasado se han presentado soluciones de conectividad basadas en shield Ethernet, WiFi, Bluethooth, etc., pero todas tenían que hacer frente a la vocación "microcontrolador" de los dispositivos y con los limitados recursos de memoria disponibles. Era casi imposible gestionar distintos usuarios y páginas HTML complejas, por no

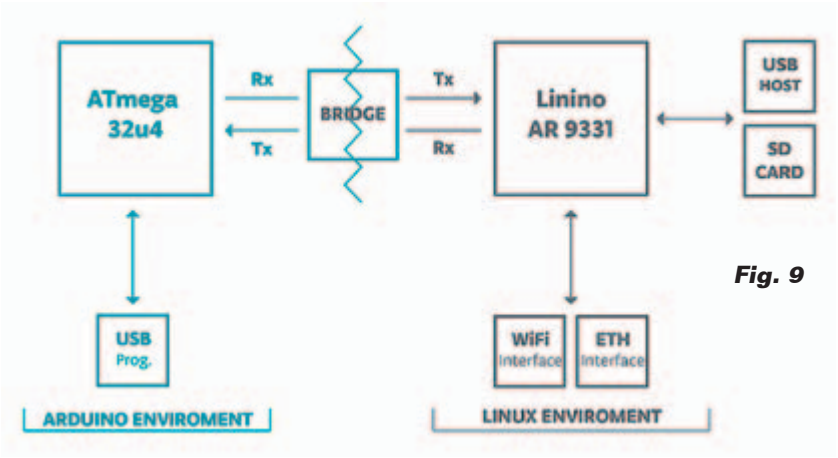


Fig. 9

hablar de la gestión de las concurrencias entre accesos web y gestión de sensores y periféricos de E/S. Con esta arquitectura se ha superado todo esto y cada sección se ocupa mejor de su respectiva funcionalidad. La disponibilidad de librerías y ambientes de desarrollo adecuados a gestionar el "puente" de comunicación integrado en el IDE y desarrollo en la lógica de simplificación propia de la "filosofía" Arduino, hacen el resto, posibilitando el desarrollo de soluciones, incluso complejas, con la habitual sencillez y linealidad.
Paremos un momento en la solución Bridge: los procesadores ATmega 32u4 y Atheros AR9331 se conectan mediante puerto serie; del lado de Arduino, la librería "Bridge" permite realizar el "puente" hacia el ambiente Linino, utilizando el puerto serie Serial1, que no puede utilizarse para otros propósitos. Simplificando mucho, la librería "Bridge" envía sobre el puerto serie "comandos" de "consola" a Linino. Los "comandos" permiten activar del lado Linino los procesos capaces de hacer de puente con los servicios y aplicaciones "GNU/Linux". Del lado de Linino, el puerto serie es asignado, en el momento del boot, a la interfaz CLI (Command Line Interface). De esta manera todos los paquetes de datos de comandos enviados desde la librería "Bridge" se interpretan por Linino como comandos de shell, que activan programas en Python que se encargan de realizar las diferentes modalidades de interfaz disponibles. De esta manera es posible instaurar la comunicación entre la sección Arduino y Linino, simulando una emulación de terminal remoto (Consolas), como shell para la gestión de comandos y procesos remotos, como puente para leer y escribir archivos sobre la SD-Card de Linino desde los sketch Arduino, como puente de comunicaciones de URL HTTP para gestionar desde el sketch las peticiones pasadas del web server uHTTPd de Linino (micro Hypertext Transfer Protocol Daemon) y en otras modalidades.
Siempre del lado de Linino, están disponibles distintos instrumentos para utilizar esta tipología de comunicación. Citamos el framework REST (REpresentational State Transfer) que permite la interfaz entre los pines de E/S del lado Arduino mediante URL HTTP del tipo:

http://arduino.local/arduino/digital/13/1

que significa: lleva a nivel alto el pin de E/S digital 13. En próximos artículos presentaremos ejemplos de aplicaciones basadas en este framework. Otro framework gestionable con la librería "Bridge" es *Spacebrew*, que permite a diferentes dispositivos Client comuni-

Fig. 10

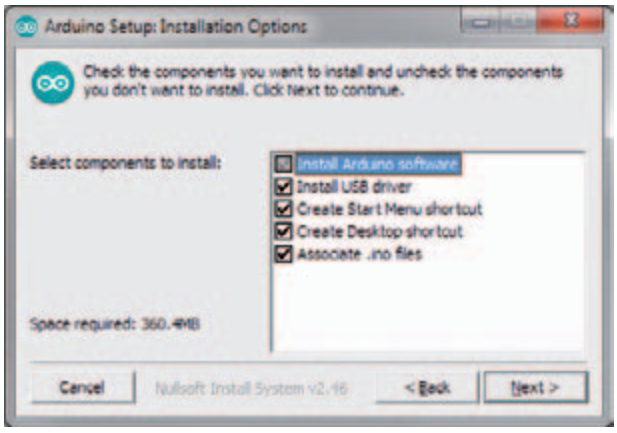


Fig. 11

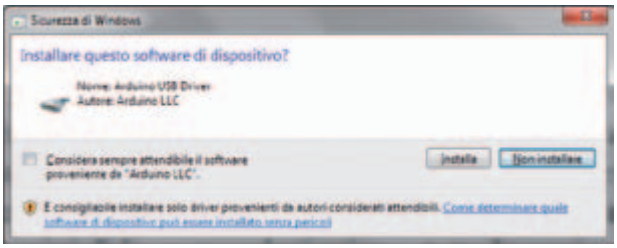
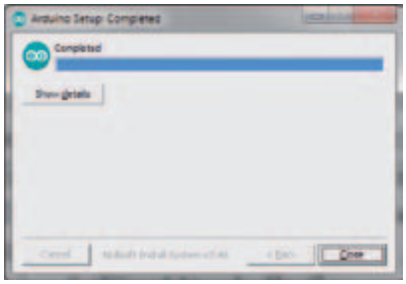


Fig. 12



carse entre ellos en modo Publisher o Subscriber por medio de un server "coordinador" público o privado. Por fin, existe la posibilidad de la interfaz de las API de Temboo que permiten comunicar con una gran cantidad de servicios "web" como las redes sociales, las educativas, el Khan Academy, los servicios de transporte, las médicas e infinitas otras. En este artículo, como primera muestra de uso, os proponemos un ejemplo de uso de las API Temboo con la librería "Bridge" con el objetivo de enviar Tweets en función de los eventos procedentes de las E/S de Arduino. Para realizar esto, debemos preparar un sketch para compilar y transferir sobre la memoria flash de la sección Arduino de Yún. Aprovechamos para introducir el uso del IDE con la Yún conectada a través red Ethernet.

EL IDE DE ARDUINO

Hemos anticipado ya que podemos descargar el IDE de desarrollo para Arduino Yún del sitio web de Arduino. Es necesario descargar los dos archivos *arduino-1.5.4-r2-windows.exe* y *arduino-1.5.4-r2-windows.zip*, o, si existen, las sucesivas versiones. El primer archivo permite la carga de los drivers USB necesarios para reconocer la tarjeta Yún e instalar el segundo paquete. Ejecutamos el primer archivo, aceptamos los términos de licencia y seleccionamos todas las opciones como en la **Fig. 10**. Respondemos con "Install" a las peticiones de las pantallas sucesivas (**Fig. 11**) y al final confirmamos con "Close" (**Fig. 12**). Terminada la instalación lanzamos el IDE desde el icono del desktop o desde el menú "Start", con lo que se abrirá la interfaz del IDE, que contiene algunas novedades. En la **Fig. 13** vemos la configuración del puerto de comunicación entre el IDE y la tarjeta Yún; en realidad se presenta la dirección IP de la tarjeta: lo seleccionamos y activamos el canal de comunicación, sin necesidad de conectar cables USB e instalar driver. Tenemos presente que este método funciona, obviamente, también si la tarjeta está conectada en modo inalámbrico, lo que constituye una gran novedad respecto al pasado, permitiendo programar y configurar la Yún aunque esta se encuentre en un lugar recóndito o montada sobre un dispositivo móvil, come ejemplo un robot. Hecha la elección del puerto de comunicación, nos queda configurar el tipo de dispositivo - obviamente Arduino Yún -, a través de la ruta "Instrumentos">>"Board". El resto corresponde a lo que hemos hecho siempre con las otras tarjetas Arduino.

LA INTERFAZ TWITTER

La aplicación que os proponemos aquí está basada en uno de los ejemplos proporcionados con el IDE y permite, utilizando las librerías y el servicio Temboo,

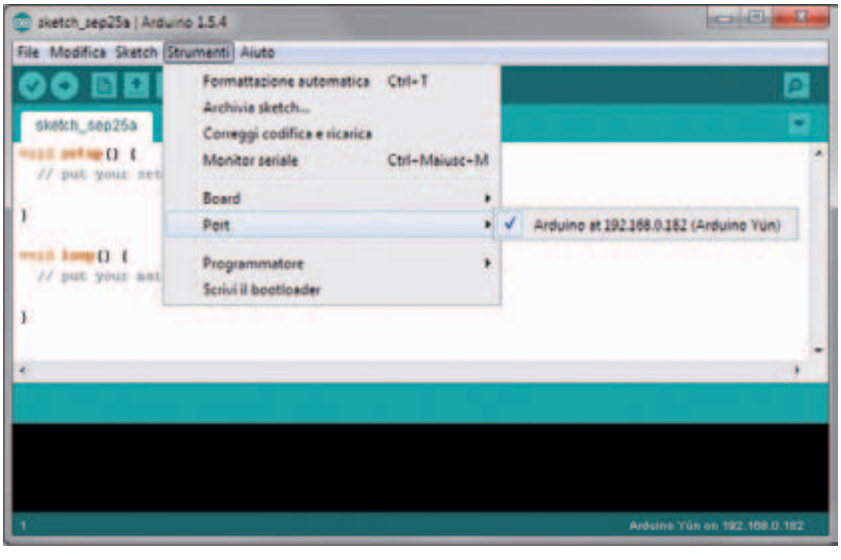


Fig. 13

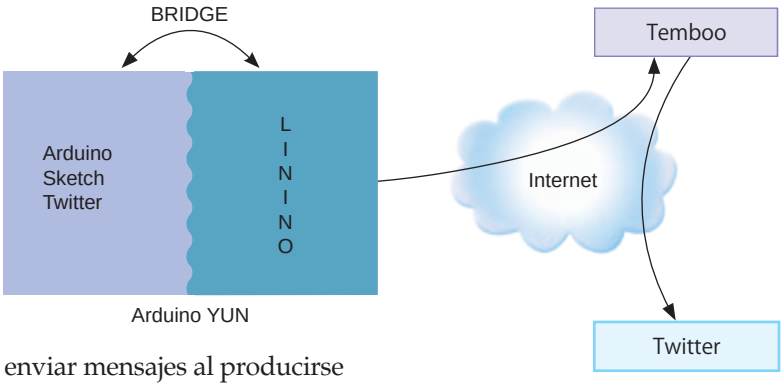


Fig. 14

enviar mensajes al producirse un evento local, que en nuestro caso es la presión de un pulsador conectado al pin 5 de Arduino Yún. Para obtener este resultado debemos realizar una cadena de comunicación como la que podéis ver en la **Fig. 14**. El servicio Temboo ofrece la posibilidad de conectarse para enviar y recibir mensajes de y para distintos tipos de servicios disponibles en red, entre los cuales las redes sociales, servicios de pago como PayPal, servicios meteorológicos y médicos, de formación y muchos otros, para un total de más de cien clases de API web disponibles. Simplificando mucho se conecta a Temboo, se instancian objetos utilizando las clases puestas a disposición por el mismo Temboo y, mediante este mecanismo, preguntamos a Temboo para enviar o recuperar informaciones de o hacia un cierto servicio.
El registro gratuito permite enviar/recibir 1.000 mensajes al mes; para gestionar más es necesario una suscripción de pago.
El programa que "gira" sobre la sección Arduino predispone la conexión hacia el servicio Temboo, que incluye tanto las credenciales de acceso al mismo servicio, como a la aplicación Twitter a la que se quiere enviar el mensaje. En cuanto a Temboo, en la sección "setup" es instanciado el objeto necesario para la conexión con Twitter. Al termino del setup, el programa entra en el bucle principal, donde en cada ciclo se verifica si el pulsador ha sido pulsado o no. Si no ha sido presionado, el programa se prepara para un nuevo ciclo; en caso contrario, se inicia el proceso de envío del mensaje Twitter. Esto conlleva la compilación de los datos del objeto con el ajuste del mensaje mismo, de las credenciales de acceso a Temboo y Twitter, y finalmente con la invocación del método "run" del objeto que procede a la ejecución del proceso verdadero y propio de envío del mensaje. En este punto, el programa se suspende hasta recibir el estado de retorno de la ejecución, con el informe de cualquier anomalía.
Para tener bajo control el comportamiento del programa hemos añadido dos LED: uno verde que queda encendido después de presionar el pulsador durante todo el tiempo necesario para enviar el Tweet, y uno rojo que se enciende parpadeando de manera intermitente en caso que se verifique algún error, continuando en este estado hasta que el error se resuelve. Para realizar el circuito simple de nuestra aplicación, hemos utilizado el shield experimental para Arduino con la breadboard de fácil inserción. Ambas tarjetas están disponibles en Futura Elettronica, con los códigos 7300-PCBPROTOV10 (más 1 7300-STRIP6 y 3 7300-STRIP8) y 7300-BBMINIW.
En la **Fig. 15** veis el prototipo, y su esquema de cableado está dibujado en la **Fig. 16.**

CONFIGURACIÓNTWITTER Y TEMBOO

Para realizar este ejemplo es necesario disponer de cuenta Twitter a la cual enviar los Tweets y configurar una aplicación específica de la cual utilizaremos las credenciales de acceso al interior de nuestro sketch. Además es necesario registrarse en Temboo para obtener las credenciales necesarias para utilizar el servicio y configurar nuestro sketch también con estas.
Empecemos con Twitter: si no tenemos cuenta Twitter creamos una, siguiendo las instrucciones indicadas en el proceso de registro. Una vez obtenido el password (vía e-mail) vamos a la dirección *https://dev.twitter.com/apps/new*, donde, si no estamos logeados ya, nos pedirán el userid y password de Twitter (**Fig. 17**).
En la página que nos aparece (**Fig. 18**) creamos una aplicación rellenando los campos obligatorios. En cuanto al campo "website" podemos proporcionar una dirección web inventada, mientras sintác-

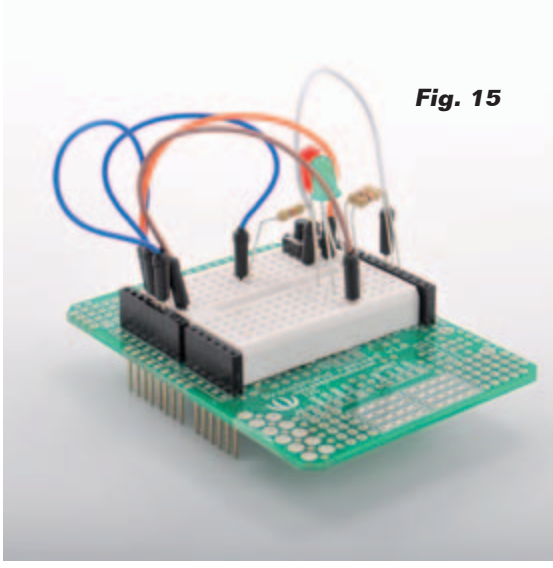


Fig. 15

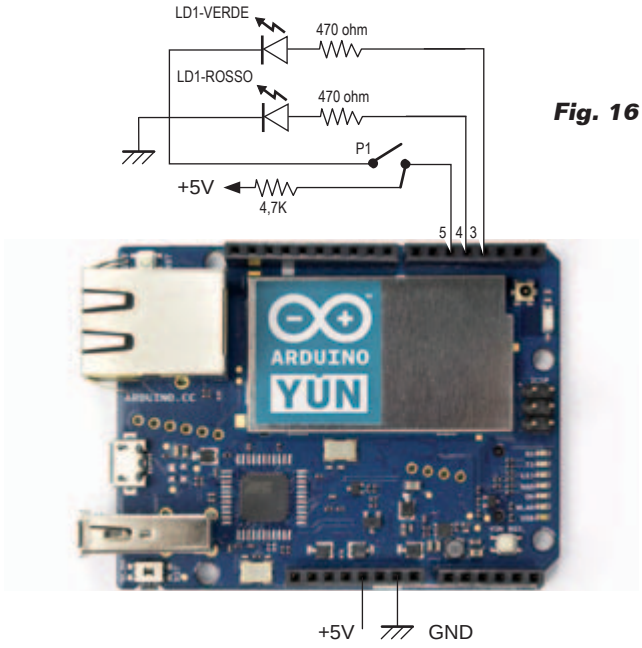


Fig. 16

Fig. 17

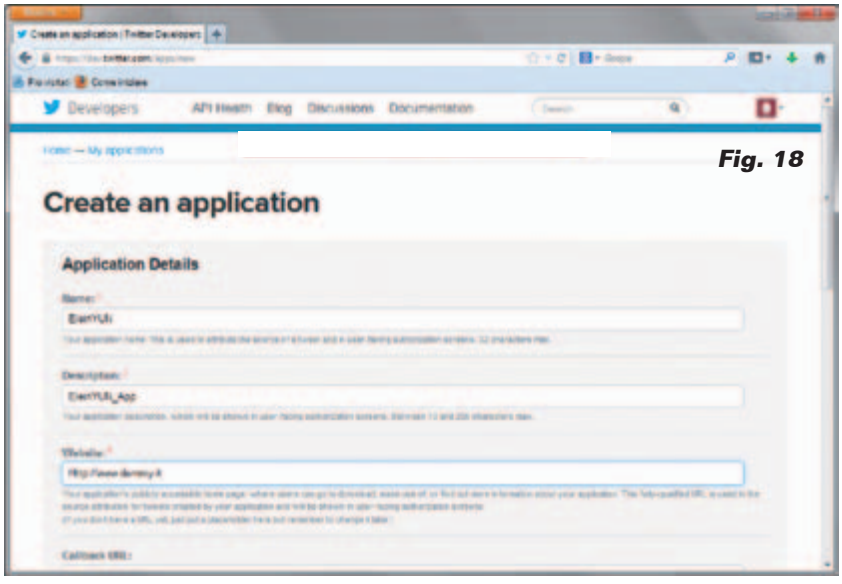


Fig. 18

ticamente sea correcta: tipo *www.noesunsitio.com.*
Rellenamos el campo captcha y confirmamos haciendo clic sobre el pulsador azul del final de la página. En la página "settings" de la **Fig. 19**, en la sección "Application Type" seleccionamos la opción "Read and Write". Confirmamos también esta página y pasamos a la siguiente, que se puede ver en la **Fig. 20**, donde nos proporcionan la primera serie de códigos que tendremos que insertar en nuestro sketch. Se trata, en particular, de los campos "Consumer key" y "Consumer secret". En la sección "Your access token" pulsamos el botón azul para crear una copia de token de acceso.
De esta sección nos servirán los campos "Acces token" y "Access token secret". Mientras tenemos abierta esta página, abrimos una nueva pestaña en el browser y vamos a la dirección de registro del servicio Temboo (**Fig. 21**) *https://temboo.com/signup*; insertamos nuestro e-mail en el único campo disponible y atendemos a la confirmación por parte de Temboo. Una vez recibido el mail pulsamos sobre el link proporcionado y, en la página de confirmación del registro, seleccionamos el usuario y un password que utilizaremos para el login en la página (**Fig. 22**) en la dirección *https://temboo.com/login*.
Insertamos todos los campos requeridos y entramos en la página de nuestra cuenta, como se ve en la **Fig. 23**; aquí seleccionamos el elemento de menú "APPLICATION KEYS" y una vez dentro, clicamos sobre la opción "(show)" pegada a la fila de asteriscos bajo el epígrafe "KEY". De esta manera obtenemos los últimos datos necesarios para personalizar nuestro sketch: en la práctica, el equivalente de userid y password necesarios para usar las librerías de las

clases y las API disponibles por el mismo Temboo. De hecho, un tipo de servicio de "consumo" donde un cierto número de usos al mes es gratuito, mientras que después es necesario una suscripción de pago.

EL SKETCH ARDUINO

Pasemos ahora a describir el funcionamiento del sketch para Arduino (**Listado 1**) cuya riqueza de comentarios no debería dejarnos dudas sobre el funcionamiento. Copiamos el código en el IDE, teniendo en cuenta que es siempre posible descargarlo de la página web *www.nuevaelectronica.com*. Ahora copiamos y pegamos los campos de las credenciales de la página web de Twitter, que hemos dejado abierta, y Temboo, en los campos relativos del sketch. Sugerimos el método de copiar y pegar porque debido a la complejidad de la clave, en caso de copia manual, puede llevar con certeza a cometer errores difícilmente identificables. Hemos elegido la modalidad upload del código sobre Arduino mediante ethernet, renunciando a usar el USB. Configuramos correctamente "Board" y "Port" y hacemos clic sobre el pulsador "Carga": primero ejecutará la compilación del código y a continuación ejecutará el código compilado sobre la tarjeta Yún. Eso es todo. Cuando termine pulsamos el botón sobre la breadboard: el LED verde quedará iluminado el tiempo necesario para el envío del Tweet a nuestra cuenta; cuando termine se apagará y será posible pulsar el botón de nuevo. Si ocurre un error se enciende también el LED rojo y vuelve a intentar el envío. Para ver qué ocurre, podemos provocar el error desconectando el cable de red después de haber pulsado y después de un poco volvemos a conectar el cable y vemos que, al

Fig. 19

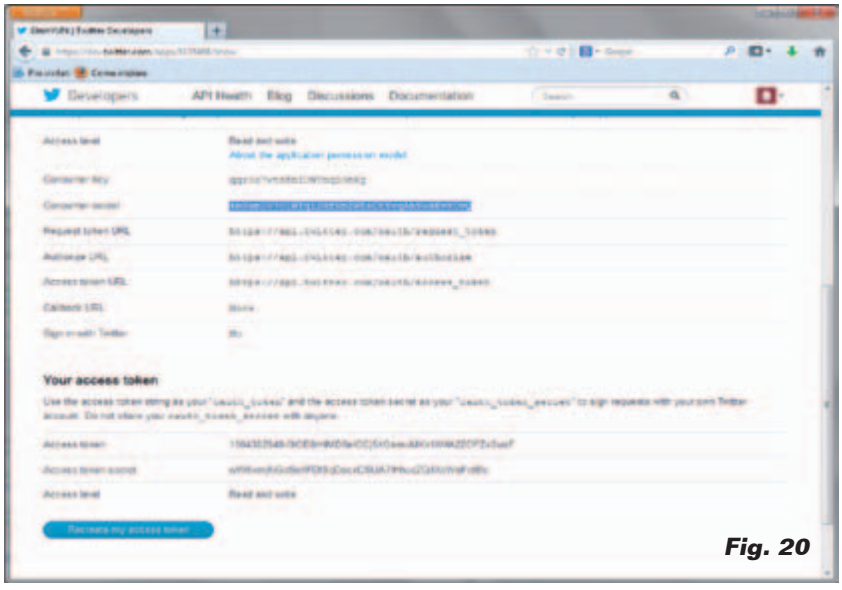


Fig. 20

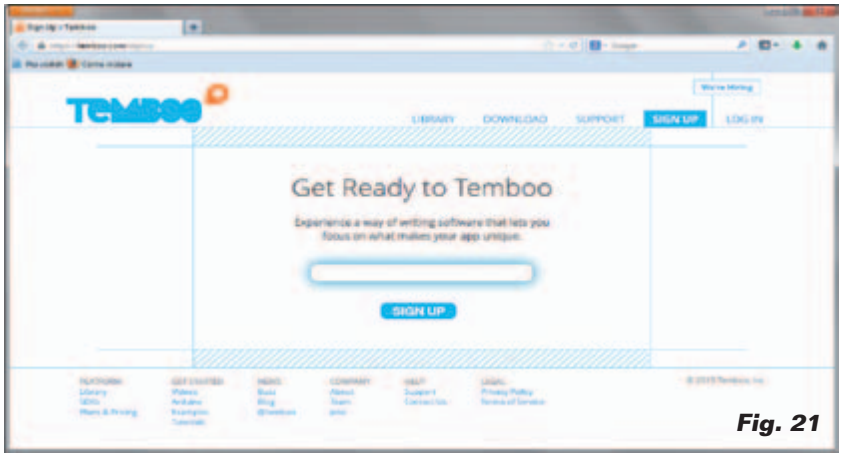


Fig. 21

Listado 1

```
/*
  Envía un Tweet
*/

#include <Bridge.h>
#include <Temboo.h>

/*** SUSTITUIR LAS CREDENCIALES  ***/

const String TWITTER_ACCESS_TOKEN = "1904382948-0....v1XhfA2ZOFZxSuxF";
const String TWITTER_ACCESS_TOKEN_SECRET = "wW5xmjNGoSoX......7IHhucZQ4XoWoFotBc";
const String TWITTER_CONSUMER_KEY = "qqr8c7wt.....qD0eXg";
const String TWITTER_CONSUMER_SECRET = "4bNamlV7U1......eCtYvqAhSwsKvH0vQ";

const String TEMBOO_ACCOUNT = "dev.....cain";  // your Temboo account name
const String TEMBOO_APP_KEY_NAME = "myF....pp";  // your Temboo app key name
const String TEMBOO_APP_KEY  = "1b.....32-9";  // your Temboo app key

int numRuns = 0;   // Contadores para gestionar el envío en caso de error
int maxRuns = 1;

int led_rojo = 4; // LED error en curso
int led_verde = 3; // LED envio TWEET en curso
int buttonPin = 5; // Pulsador de envío TWEET
int buttonState = 1;

void setup() {
  pinMode(led_rosso, OUTPUT);
  pinMode(led_verde, OUTPUT);
  pinMode(buttonPin, INPUT);
  Bridge.begin();    // Inicializa la comunicación con LINUX vía serie compartida
}

void loop()
{
   buttonState = digitalRead(buttonPin);
   if (buttonState == LOW) {              // Se pulsa el botón

      while (numRuns < maxRuns) {         // intenta los envois hasta buen puerto

        // Texto del mensaje TWEET
        String tweetText("Alguien ha pulsado el botón - "  + String(millis()) );

        digitalWrite(led_verde, HIGH);

        TembooChoreo StatusesUpdateChoreo;  // Instancia el objeto para envío TWEET

        // Invoca el client Temboo
        StatusesUpdateChoreo.begin();

        // Pone las credenciales para acceder a Temboo
        StatusesUpdateChoreo.setAccountName(TEMBOO_ACCOUNT);
        StatusesUpdateChoreo.setAppKeyName(TEMBOO_APP_KEY_NAME);
        StatusesUpdateChoreo.setAppKey(TEMBOO_APP_KEY);

        // Pone la librería de API a utilizar (Twitter > Tweets > StatusesUpdate)
        StatusesUpdateChoreo.setChoreo("/Library/Twitter/Tweets/StatusesUpdate");

        // Pone las credenciales de acceso a Twitter
        StatusesUpdateChoreo.addInput("AccessToken", TWITTER_ACCESS_TOKEN);
        StatusesUpdateChoreo.addInput("AccessTokenSecret", TWITTER_ACCESS_TOKEN_SECRET);
        StatusesUpdateChoreo.addInput("ConsumerKey", TWITTER_CONSUMER_KEY);
        StatusesUpdateChoreo.addInput("ConsumerSecret", TWITTER_CONSUMER_SECRET);

        // referencia al mensaje a enviar
        StatusesUpdateChoreo.addInput("StatusUpdate", tweetText);

        // Requiere la ejecución del proceso de envío
        unsigned int returnCode = StatusesUpdateChoreo.run();

        // El código de retorno (0) significa proceso llegado a buen puerto
        if (returnCode == 0) {
           numRuns = numRuns + 1;
           digitalWrite(led_verde, LOW);
        } else {
           // En caso de error non se sale del loop while y se reintenta el envío
           while (StatusesUpdateChoreo.available()) {
             char c = StatusesUpdateChoreo.read();
             }
        digitalWrite(led_rosso, HIGH);
        delay(5000);
        digitalWrite(led_rosso, LOW);
        delay(5000);
        }
      StatusesUpdateChoreo.close();
      }
   numRuns = 0; // Todo ok en espera de una nueva pulsación del botón
   }
}
```

final, nuestro Tweet llegará a su destino (**Fig. 24**).

CONCLUSIONES

En este primer acercamiento a la tarjeta Arduino Yún hemos tocado solo superficialmente su potencial, sobretodo en el plano didáctico. Yún se presta de hecho como banco de prueba, simple y poco costoso para profundizar los temas relativos al mundo GNU/Linux y a las redes, además de las propias del mundo Arduino, como ya era posible con sus predecesores. El sistema operativo Linino, con su corazón OpenWRT, se presta a experimentar, a bajo coste, la configuración y gestión de redes desde el punto de vista de la administración y de la seguridad, de manera cableada o wireless. Con un puñado de tarjetas Yún se pueden realizar configuraciones de red complejas; y esto, sobre todo en los tiempos de dificultades económicas en los que vivimos, debería interesar decididamente a las Escuelas Superiores y la Universidad. La misma cosa se puede decir para el estudio de aplicaciones embebidas, típicamente compuestas por secciones dirigidas al muestreo y elaboración de datos desde y hacia sensores y E/S analógicas y digitales y de secciones de coordinación e interfaz humano haciendo uso de soluciones de aplicación basadas en web. Todo ello llevado a un nivel simplificado, propio de la filosofía "Arduino", que permite también y sobre todo, a quien es principiante encontrar una satisfacción inmediata y un empujón para profundizar en las tecnologías que cada vez estarán más presentes en nuestro entorno. No dejaremos de profundizar las posibilidades de Yún y de proponéroslas en próximos artículos, junto a las demás novedades que están surgiendo. *(180043)*

Fig. 22

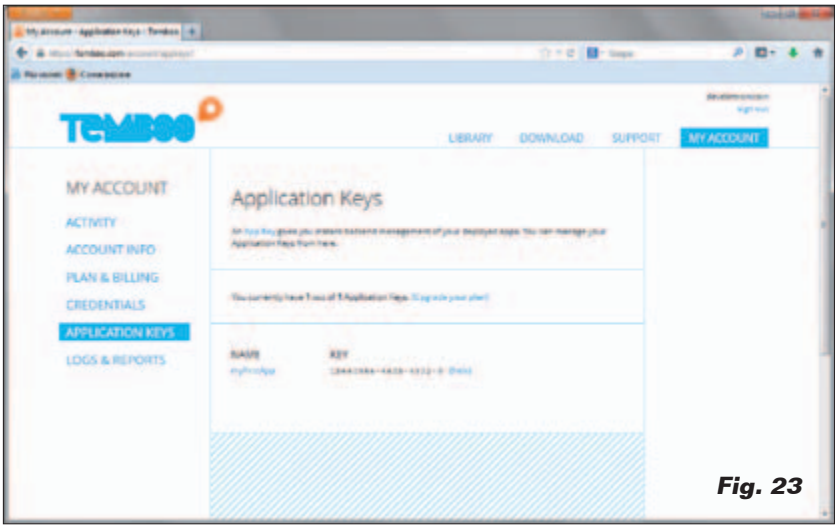


Fig. 23

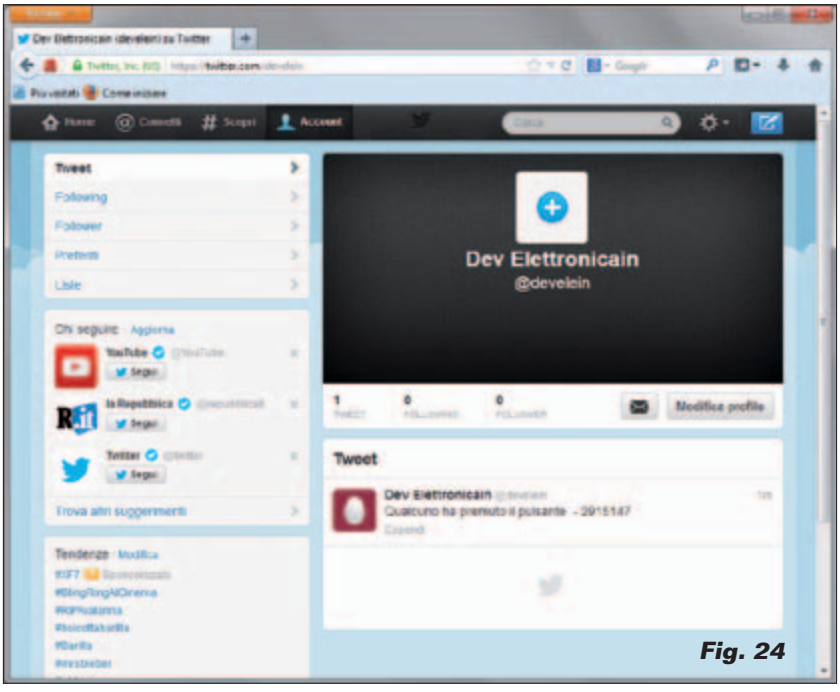


Fig. 24

la tienda Nueva Electrónica 3.0

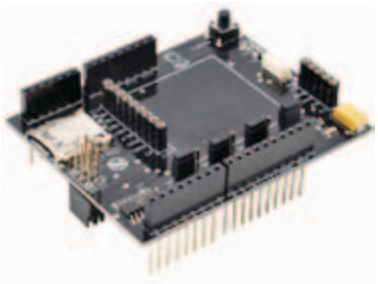
En Nueva Electrónica queremos facilitarte las cosas, por eso en nuestra tienda encontrarás los kits y módulos que te mencionamos en la revista además de una amplia gama de productos seleccionados para ti. Cada semana encontrarás algo nuevo para completar tu laboratorio o para llevar a cabo esa aplicación que tienes en mente: Instrumentación, plataformas de desarrollo para sistemas embebidos, shields para las plataformas más populares, kits, herramientas ... todo lo que necesitas está en www.nuevaelectronica.com.

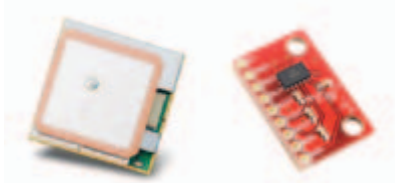
3DRAG: La impresora 3D

Impresora 3D versión 1.2, capaz de fabricar objetos de tamaño máximo de 20 x 20 x 20 cm utilizando filamento de ABS o PLA de 3 mm. Impresiones extremadamente rápidas y precisas, incluso a altas velocidades. Compatible con todo el software y el firmware de RepRap disponible gratuitamente, permite crear objetos en plástico de cualquier forma.
Fabricada con perfiles de aluminio diseñados para poder montarse a presión que ofrecen ligereza y rigidez mecánica para la supresión de vibraciones y resonancias no deseadas.
La impresora está disponible en kit para montar y también completamente montada y comprobada. Ahora también disponibles rollos de ABS y PLA. Consulta la web.

Impresora 3DRAG en **Kit** (7350-3DRAG-K): **580,00 €**
Impresora 3DRAG **Montada** (7350-3DRAG-M): **760,00 €**

Shield GPS Para Arduino

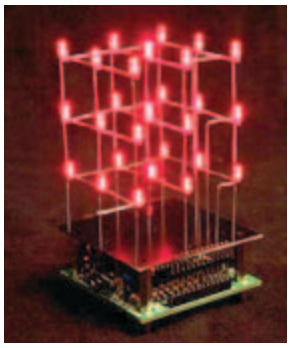




Con este shield puedes añadir un GPS a tu Arduino para que sirva de orientación a tus sistemas móviles, robots, seguimiento ... Este módulo dispone de espacio y conector para alojar el receptor GPS (ref. EM406A, no incluido) y de zócalo para el acelerómetro de tres ejes (ref. MMA7361, no incluido) que son precisos para llevar a cabo la aplicación descrita en esta edición; un sistema que registra las posiciones recorridas por el sistema mientras esté en movimiento.

Shield GPS (7100-FT1017M): **22,00 €**
Receptor GPS con antena integrada (8160-EM406A): **45,00 €**
Módulo acelerómetro de 3 ejes (7300-MMA7361): **14,00 €**

Cubo Luminoso



Construye un cubo luminoso con 3 planos y 9 columnas (27 diodos luminosos en total) manejado por un microcontrolador y provisto de interfaz USB para permitir a un ordenador personal, en el que funciona un software específico, ordenar los efectos luminosos a través de una pantalla de control simple e intuitiva.

Kit completo (8220-MK193): **28,00 €**

Arduino Battery Shield

Optimizamos la autonomía de los sistemas Arduino alimentados por batería, gracias a un temporizador basado en RTC con función despertador programable.

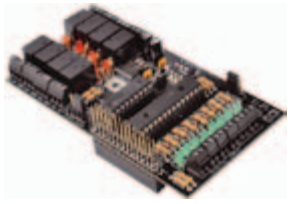
Kit completo (8190-BATTERYSHIELD): **19,50 €**

Controladores Arduino: Uno, Mega, Yún

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares.
Elige la que más se ajuste a las necesidades de tu aplicación; entradas/salidas, conectividad, memoria, ...

Arduino UNO Rev.3: **24,50 €** Arduino MEGA 2560 Rev.3: **51,00 €** Arduino YÚN: **63,00 €**

Shield I²C de expansión E/S para Raspberry Pi



Shield para Raspberry Pi basado en el integrado MCP23017 que permite aumentar el número de entradas/salidas digitales. Permite disponer de ocho entradas y otras tantas salidas digitales. El estado de las ocho entradas viene representado por un LED para cada una. Cada una de las ocho salidas controla un relé al que se pueden conectar cargas externas.

Kit completo (7100-FT1093K): **32,00 €**

Interfaz de 4 canales E/S Bluetooth Android Based

Basado el nuevo módulo Bluetooth RN-42 de Roving Network, esta tarjeta de 4 canales se convierte en un sistema de telecontrol basado en Android (Android Based). Se trata de una tarjeta de gestión de entradas/salidas provista de cuatro salidas a relé y otras tantas entradas opto-aisladas a nivel de tensión, controlables a través de Bluetooth.

Kit completo (7100-FT1095K): **64,00 €**






VISNOC TECHNOLOGY



INVESTIGACION E INNOVACION TECNOLOGICA

VISNOC TECHNOLOGY, S.L.

Polígono Industrial Las Salinas de Poniente c/ Alfred Nobel nº 22
11500 El Puerto de Santa María (Cádiz) - España
http://www.visnoc.com info@visnoc.com
Tlf. 956 144 424 - Fax. 956 548 241

Unión Europea
Fondo Social Europeo
"El FSE invierte en tu futuro"