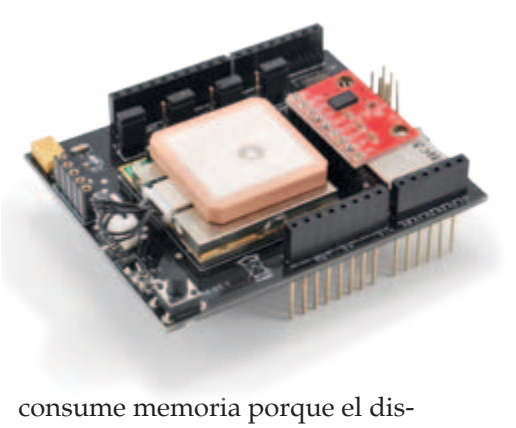
GPS LOGGER SHIELD

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.

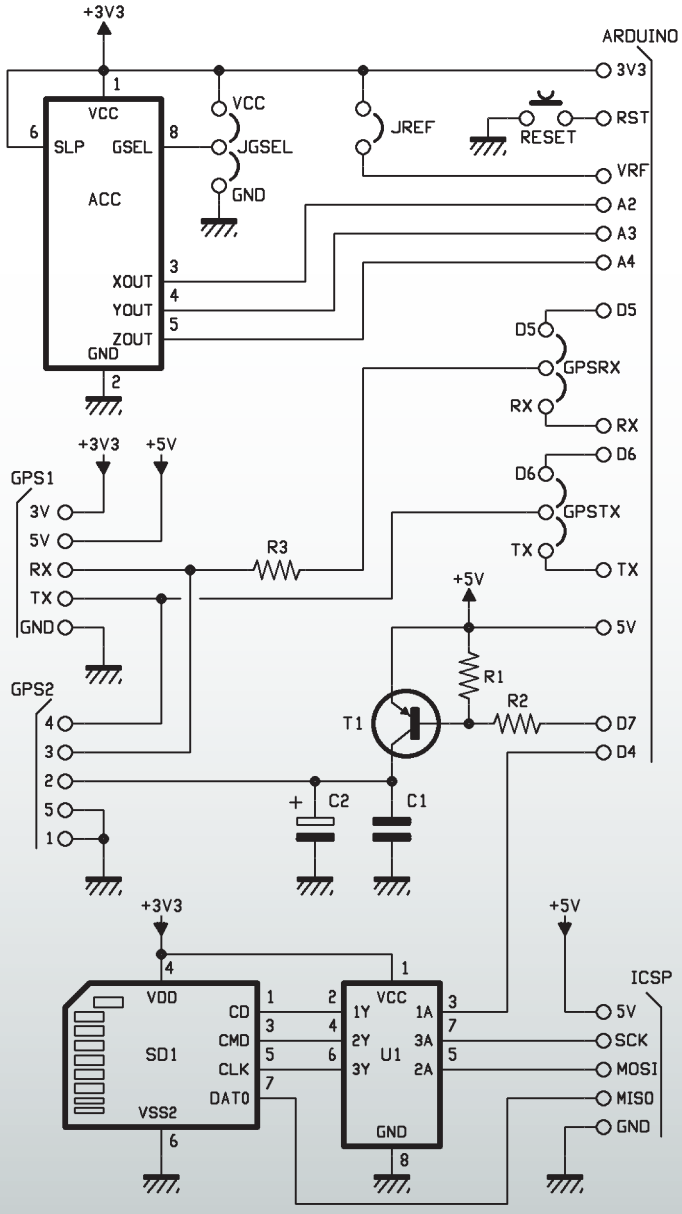
LABORATORIO INFORMATICO
FREE ED OPEN

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

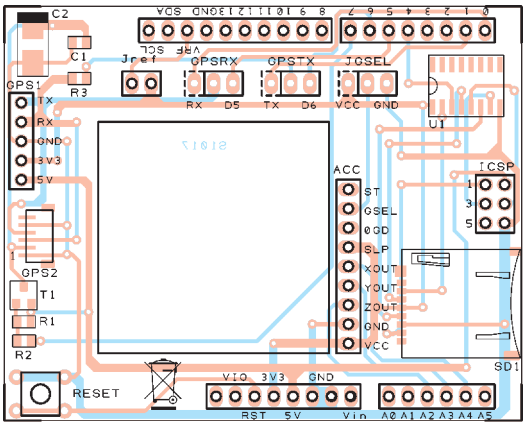
El logger que os proponemos consta de un shield, sobre el cual se encuentra principalmente el receptor GPS, el lector de SD-Card y el acelerómetro, además de una tarjeta Arduino UNO; hay que señalar que el receptor se conecta mediante dos conectores (alternativos el uno del otro), con lo que nada impide 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.

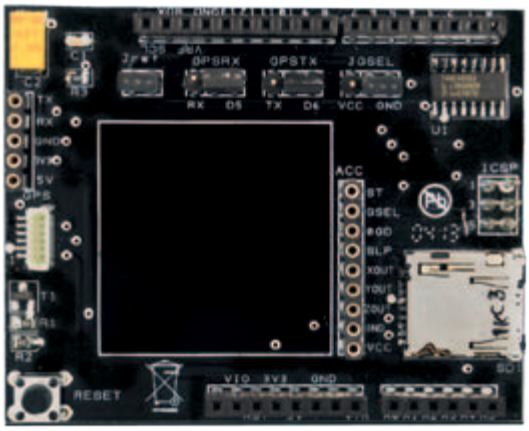
Veamos entonces en qué consiste el circuito del shield, que gracias a la reducida absorción de corriente, se alimenta directamente desde el Arduino. En nuestras mediciones, alimentando Arduino a 5 V, indican consumos de alrededor a 35 mA con el GPS activo, reducido a cerca de 5 mA desactivando el GPS 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**]





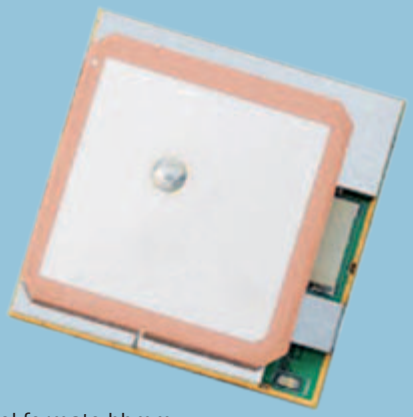
Lista de materiales

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

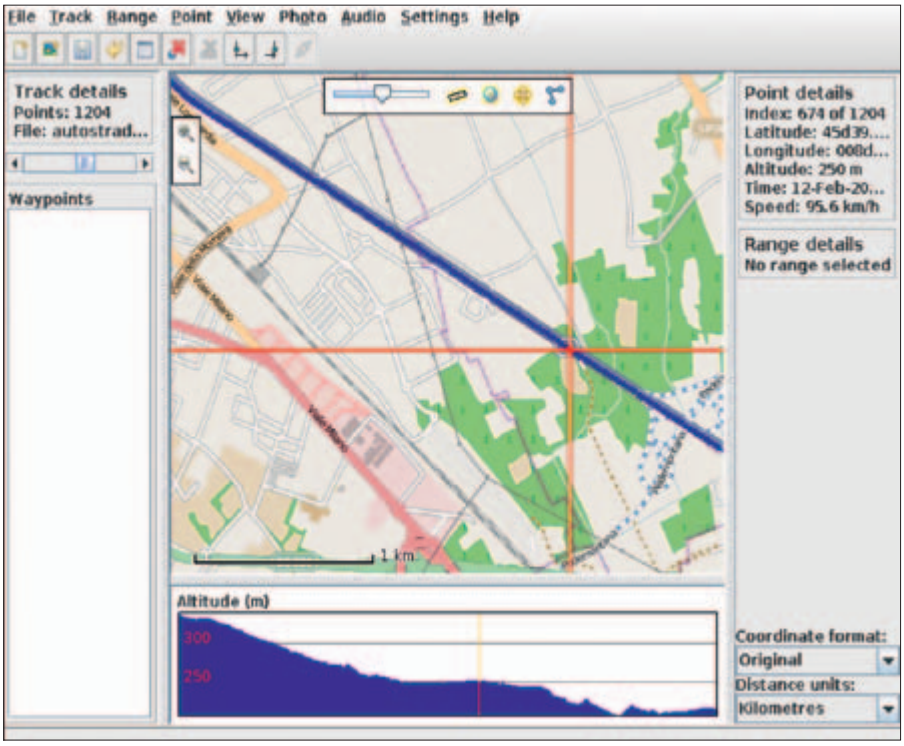
El estándar **NMEA0183**



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

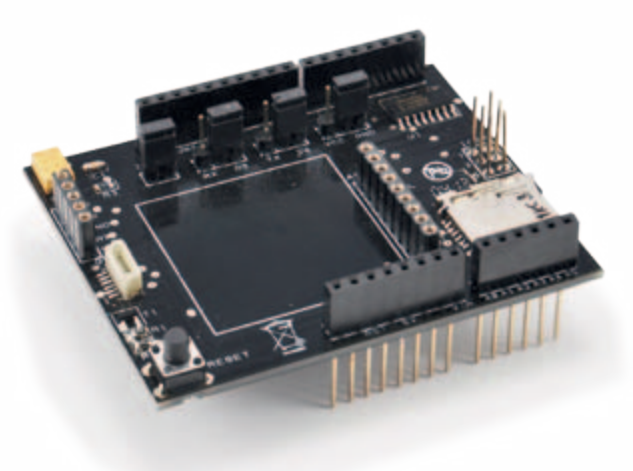
La construcción del shield es bastante simple: una vez obtenida la placa (los archivos de diseño están disponibles en nuestra pá-

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