LEWE
PULSERA BIOMÉTRICA

Cuando se lleva en la muñeca, detecta la temperatura y la sudoración y transmite la información a través de bluetooth a un smartphone, conectado a Internet y preparado para publicarlos en un servicio especial en la nube.

MIRCO SEGATELLO
y ALESSANDRO PASQUALINI

El desarrollo de los medios de comunicación ha experimentado un impulso increíble en los últimos años; en particular, la telemática ha alcanzado niveles cada vez más sofisticados, dando lugar a una verdadera revolución en la difusión de la información, rompiendo barreras económicas, geográficas y culturales. El uso de las nuevas tecnologías garantiza hoy la oportunidad de hacer circular las ideas y proyectos, dando vida a un flujo de comunicación donde los argumentos y contenidos digitales son compartidos por millones de usuarios en todas las partes, son al

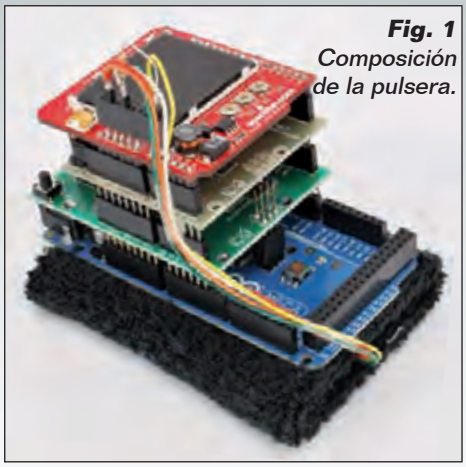

Fig. 1
Composición de la pulsera.

mismo tiempo usuarios, creadores y coautores de un trabajo global. El proyecto Lewe parte de este ámbito y los objetivos de recogida y reprocesamiento de las tecnologías disponibles para lograr una idea a bajo coste aplicable a todos los sectores que utilizan sensores para la recolección de datos. El proyecto tiene como objetivo la realización de una portal en la nube, llamado Lewe, el protocolo de comunicación, denominado jack (capaz de transportar los datos de una manera fiable) y de una serie de dispositivos de diferente naturaleza que explotan esta estructura para publicar automáticamente los datos en la web. Ejemplos de la aplicación de este protocolo pueden ser la puesta en marcha de una alarma de coche, la administración remota de un invernadero, o bien el brazalete biométrico Lewe descrito en estas páginas. La pulsera biométrica nace de la idea de los autores: Alexander, que ama inmensamente la escalada y el paracaidismo, y Mirco, su entrenador personal, que quería conocer en tiempo real los datos biométricos de Alexander simplemente accediendo a la nube online de Lewe. Todo esto queda plasmado en una pulsera para la lectura de los datos biométricos que se enviarán a través de bluetooth a un smartphone Android y, a través de él, a un servicio cloud para el archivo y la organización algunos datos. En la versión experimental, éste proyecto se llevó a cabo (ver Fig. 4) mediante una tarjeta Arduino Mega que alberga un Bluetooth de Futura Electrónica, un shield RTC (siempre de Futura Electrónica) y un LCD a color de pantalla Sparkfun; claramente todo tiene un tamaño nada despreciable, pero nuestra intención es explicar cómo realizar la aplicación, sin perjuicio de que si crees que puedes conseguir tu propia arquitectura de hardware Arduino Mega (los esquemas de las Placas Arduino son de dominio público...) e integrar todo en una sola tarjeta o a lo sumo dos, muy compacta y aplicable en la muñeca mediante un recipiente adecuado con pulsara (tipo medidor de presión sanguínea). En el proyecto que hemos desarrollado se incluyeron las siguientes funciones:

- detección de la temperatura y sudoración del cuerpo;
- visualización local de los parámetros adquiridos;
- envío de los valores a una aplicación instalada en un smartphone;
- carga gratuita de los datos en una nube online;
- organización de los datos en gráficos simples que ayudan a comprender la evolución temporal de los parámetros biométricos y compararlos con los de otros usuarios.

Fig. 2
El brazalete puesto.

HARDWARE

El brazalete se ha desarrollado en torno a la plataforma Arduino Mega (Fig.3). Para la comunicación con el smartphone se ha utilizado un módulo Bluetooth RN-42 interconectado con Arduino a través de un shield Bluetooth

de Futura Electrónica, mediante el hardware serie (pin 0 y 1). En este proyecto hemos preferido utilizar un interfaz serie software en los pines 6 y 10, para permitir que el interfaz serie hardware quede libre para la comunicación con el PC, incluso durante la comunicación Bluetooh, los que resulta útil para la depuración del software. Para ello, debemos interrumpir las pistas que salen de los pines 0 y 1 y conectarlos a los pines 10 y 6 del módulo respectivamente; por comodidad, se pueden usar los pines del conector denominado RS232 ya presente en el shield.

Para evitar que los pines

0 y 1 del módulo Bluetooth entren en comunicación con los contactos inferiores de la placa Arduino, puede cortar o bien doblar hacia el exterior (Fig.5), a fin de no introducirlos en los conectores de la tarjeta. Para la lectura de la temperatura hemos utilizado un sensor común LM35 con encapsulado TO-92, cuya salida está conectada a la entrada analógica A0 de Arduino. En cambio, para la detección de la sudoración se utilizaron dos electrodos conectados

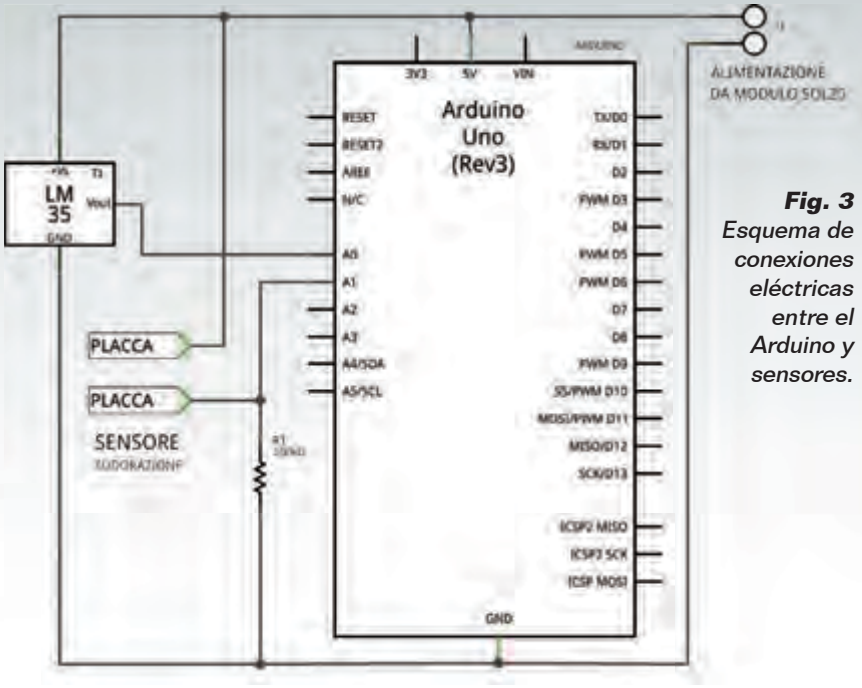

Fig. 3 *Esquema de conexiones eléctricas entre el Arduino y sensores.*

a un divisor de tensión unido a la entrada analógica A1 del Arduino. El inicio del funcionamiento del sistema de detección de la transpiración se basa en la conductividad del sudor, el cual, por ser una solución salina, permite que la corriente fluya de un electrodo al otro; cuanto mayor sea el nivel de la transpiración, menor es la resistencia medida entre los electrodos. Estos dos electrodos deben ser del tamaño de una moneda de un céntimo y, obviamente, tener buenas características de conductividad y resistencia a la oxidación. Con el divisor de tensión obtenemos un valor de tensión dependiente del nivel de sudoración, que a su vez está relacionado con los estímulos emocionales de quién se lo pone y también es un buen indicador del esfuerzo físico del atleta. Evidentemente las tensiones y corrientes implicadas son tan pequeñas como para no suponer ningún peligro para la persona que lleva la pulsera. En el esquema eléctrico se puede ver que los sensores se alimentan a través del pin 5V de Arduino, pero como la pulsera estará alimentada por batería, hemos previsto la posibilidad de controlar la alimentación de los sensores, activándola solo cuando sea necesario para realizar la medida. La alimentación del sensor LM35 se puede tomar del pin 12, mientras que el sensor de la transpiración se puede alimentar desde el pin 7. Estos pines se mantienen normalmente a nivel bajo y el software los activa sólo cuando sea necesario llevar a cabo la medida (esto permite limitar el consumo eléctrico, y alargar la vida de la batería); la corriente de salida de 40mA indicada en las hojas de características del micro controlador Atmel, es suficiente para alimentar los pines afectados por la medición. Para correlacionar temporalmente los datos detectados, hemos utilizado un módulo RTC puesto en comunicación con el Arduino a través del bus I²C, mediante los pines analógicos A4 y A5. Hemos también incluido un shield LCD Color para mostrar los datos biométricos en tiempo real y el estado de funcionamiento, además de fecha y hora; las mediciones adquiridas

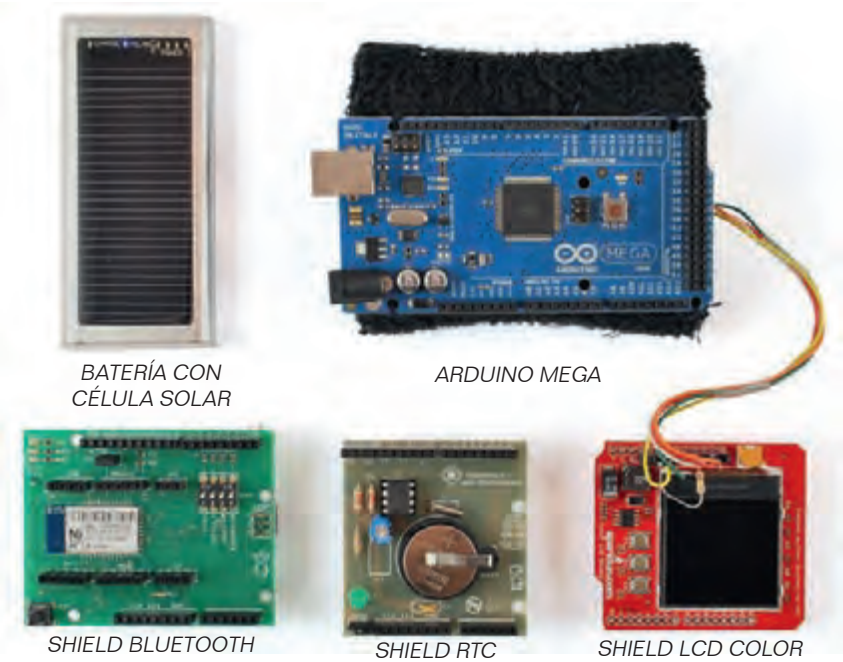

Fig. 4 - Componentes del sistema Lewe.

a intervalos regulares de tiempo ajustable, se envían en tiempo real a un smartphone para su posterior almacenamiento. Los pines 8, 9, 11 y 13 (trasferencia de datos) junto con los pines 3, 4 y 5 (botones) son utilizados por el shield LCD color. El brazalete, en ausencia de conexión Bluetooth con el smartphone, puede almacenar hasta 12 mediciones; tan pronto como se conecta, envía las 12 mediciones. Todo el sistema se alimenta mediante un módulo de batería, distribuido por Futura Electrónica (código SOL20), que le permite recargarse a través de un puerto USB, y con el pequeño panel solar incorporado, se optimiza para la utilización al aire libre. Si es necesario, el mismo módulo puede servir para recargar el smartphone. El módulo SOL20 proporciona una tensión de salida de 5V estabilizada, que se puede utilizar para alimentar el Arduino utilizando los pines +5V y GND; para hacer esto, sin embargo, es necesario cortar un cable USB, de modo que una extremo se conserva el conector USB para conectar al módulo de batería, mientras que en el otro extremo se conectan los cables directamente a los pines de alimentación +5V y GND de Arduino. Para evitar la manipulación del cable es posible usar el mismo cable utilizado para la programación del Arduino, pero en lugar de conectarlo al puerto USB del PC será insertado en el módulo SOL20: algo un poco engorroso, pero muy funcional. El mismo sistema se puede utilizar para alimentar con batería cualquier otro proyecto, con la ventaja de disponer de doble modalidad de carga: a través de USB o con un panel solar integrado.

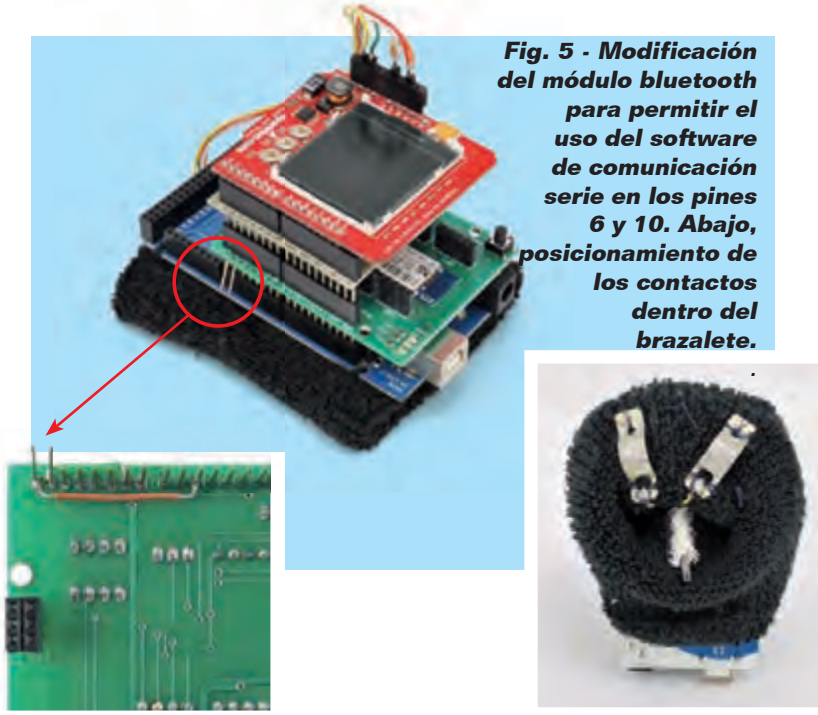

Fig. 5 - Modificación del módulo bluetooth para permitir el uso del software de comunicación serie en los pines 6 y 10. Abajo, posicionamiento de los contactos dentro del brazalete.

FIRMWARE

El firmware se ha desarrollado sobre el IDE de Arduino utilizando diferentes librerías estándar y alguna externa. La librería color LCD se utiliza para controlar el shield LCD color de Sparkfun, la librería HashMap se usa para gestionar las matrices de cadenas (strings) y, finalmente, la librería Wrapper se usa para facilitar la extracción de valores de cadena encapsulados en la matriz.
Para la utilización del shield Bluetooth se ha creado la librería Software SerialJack, basada en SoftwareSerial del SDK de Arduino, para implementar una comunicación más segura gracias a la adición del protocolo de comunicación Jack que garantiza una conexión fiable entre el emisor y el receptor, por la inclusión un sistema de acuse de recibo de los mensajes. Este protocolo se ha desarrollado para ser compatible con

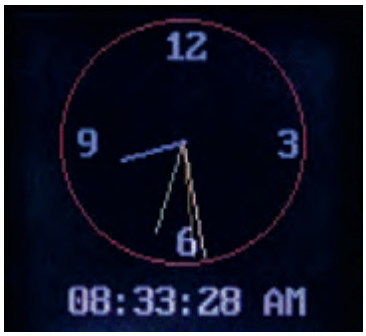

***Fig. 6** LCD mostrando el reloj..*

***Fig. 7** - LCD mostrando los datos.*

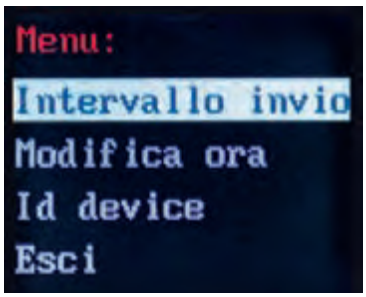

***Fig. 8** - Mostrando el menú*

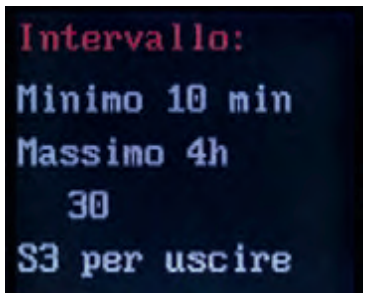

***Fig. 9** - Configuración*

cualquier medio de transmisión e implementa las siguientes funciones **boolean available (void)**, **String getMessage (void)** y void **SendMessage (String)**. La función boolean available (void) devuelve un valor booleano verdadero si hay mensajes disponibles para su descarga durante la transmisión, de lo contrario es falso. String getMessage (void) devuelve en formato cadena, una cada vez, los mensajes recibidos desde el medio de trasmisión. Por último, SendMessage (String) se ocupa de enviar por el medio de trasmisión los mensajes, como cadenas, los datos tratados por el protocolo Jack.

Éste último proporciona una clase que actúa como un contenedor para los datos llamado JData y utiliza las clases de contenedor para almacenar los datos y conservar el tipo. El método **send(JData)** del protocolo se ocupa de crear un mensaje de tipo cadena con el mensaje real para ser enviado con un identificador único del mensaje, que está compuesto por la fecha y hora de adquisición, el valor de la temperatura y el de la transpiración.

El mensaje se envía entonces al receptor para ser procesado. Una vez recibido, se selecciona la función **available()** la cual pasará el mensaje al función **getMessage()** que salvará el mensaje recibido. Para procesar el mensaje recibido se utiliza el método **execute(String)**, que comprobará si el mensaje es conforme al estándar, después extraerá el código único y los datos contenidos en el mensaje; para confirmar la recepción del mensaje, se envía al remitente el identificador único. Si el mensaje no llega o es corrupto, no habrá ningún acuse de recibo y el transmisor entenderá que el mensaje se ha perdido y lo tendrá que retrasmitir. De este modo, incluso una caída momentánea de la conexión Bluetooth no comportará la pérdida de información, que será devuelta tan pronto como se restablezca la conexión. El código de tiempo incluido en el mensaje permitirá la reconstrucción de la secuencia temporal correcta.

Para compilar correctamente el sketch de Arduino hay instalar las librerías adicionales ColorLCDShield, HashMap, Jack, RTClib, SoftwareSerialJack y Wrapper, todas ellas proporcionadas con los archivos de este proyecto; el número considerable de líneas código implica que el sketch ocupa una gran cantidad de espacio que sólo puede funcionar con una tarjeta Arduino mega 2560

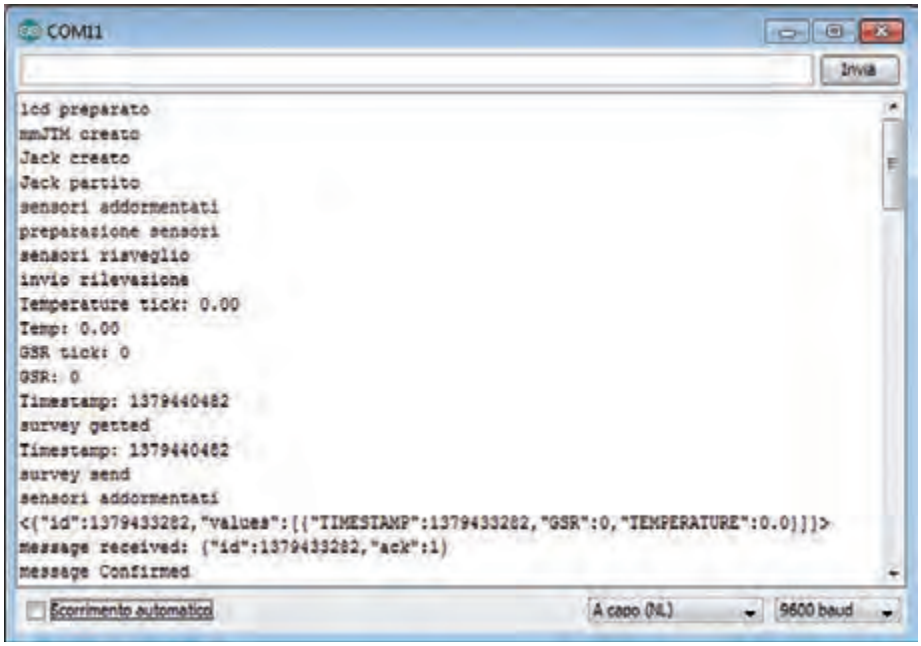

***Fig. 10** - Tráfico de datos recibidos en el monitor serie.*

que dispone de 256 KB de espacio para el programa; un Arduino UNO no dispone de suficiente capacidad para el sketch.
Durante el funcionamiento, la pulsera controla constantemente a través de USB las cadenas enviadas y recibidas por Bluetooth, de modo que pueda verificar la corrección del protocolo utilizado; el campo "id" representa el identificador del mensaje, valor clave utilizada para verificar la calidad de los mensajes recibidos. El módulo Bluetooh utiliza todos los ajustes por defecto y los puentes del shield están posicionados de la siguiente manera: AUTO=off, DEFAULT=off, MASTER=off y BAUDRATE=on. El módulo se ve obligado a trabajar como esclavo con velocidad de comunicación de 9600bps.
Si el módulo se ha programado previamente, es necesario reiniciarlo a valores de fábrica, función de la que se encarga el botón RESET situado en el módulo. Sin embargo es importante asignarle un nombre que sea compatible con "lewe" o sea del tipo LEWE_idDevice, donde IdDevice es un número de identificación del dispositivo.
Para este procedimiento hemos escrito un sketch adecuado con el que es posible asignar un nombre, distinto del predeterminado por defecto, al módulo Bluetooth.
El sketch se llama RN_42_serial - monitor.ino y permite la interconexión con el módulo RN-42 a través del monitor de serie de la placa Arduino. Después de cargar el sketch de Arduino, basta con enviar los siguientes comandos AT:
- establecer "sin salto de línea" en el monitor de serie y enviar $$$ para entrar en modo comando (respuesta: CMD);
- establecer "NL es a la vez CR" en el monitor de serie;
- enviar D para tener la configuración como una respuesta (respuesta: AOK);
- nviar SN, LEWE_01 para establecer el nombre del módulo (respuesta: AOK);
- establecer "NL es a la vez CR" sobre monitor de serie;
- enviar --- para salir del modo de comando (respuesta: END).

Dejando en ejecución éste sketch es posible interconectar con un dispositivo Android y probar un intercambio de datos simplemente instalando un APP con función de Bluetooh SPP, es decir, capaz de manejar las comunicaciones en serie a través de BT. Hemos hecho algunas pruebas con la aplicación "Bluetooth spp pro" gratuita, con la cual es posible enviar y recibir cadenas. Ésta APP también permite la modalidad "Button" con la que se pueden programar los botones para enviar cadenas predefinidas, útil para gestionar las salidas de Arduino u órdenes más avanzadas.

APLICACIÓN PARA SMARTPHONE

La aplicación para Smartphone Android constituye el verdadero corazón de Lewe, porque se ocupa de recibir los datos de la pulsera y de almacenarlos localmente en el dispositivo y, si se configura correctamente, subirlos a la nube. Se trata de una actividad principal en la que se muestran los últimos datos recibidos de la pulsera y una actividad secundaria que muestra el gráfico que contiene todos los datos almacenados. Al hacer clic sobre el icono del engranaje, se puede entrar en la configuración de la aplicación donde se selecciona el brazalete al que conectarse y se configuran los datos de acceso a la nube. La aplicación ha sido realizada en Java con el apoyo de Eclipse provisto de ADT Plugin, que permite el desarrollo de aplicaciones de Android; para los gráficos es preferible utilizar la aplicación AchartEngine, disponible bajo licencia Apache 2.0. La aplicación hace uso de cuatro servicios que trabajan en segundo plano:
- LeweService;

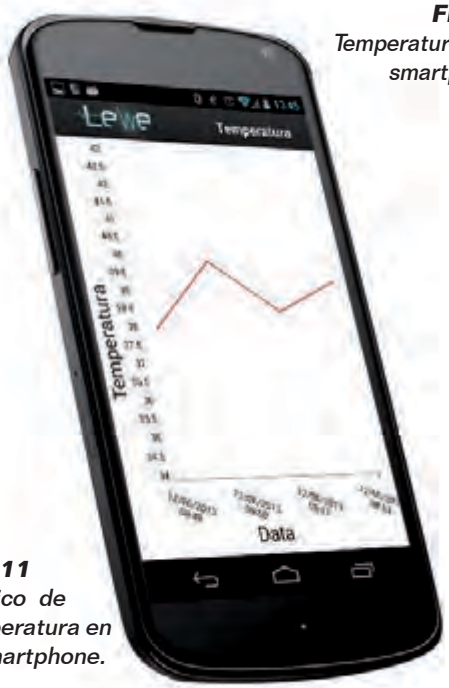

Fig. 11
Gráfico de temperatura en el smartphone.

Fig. 12
Temperatura en el smartphone.

- LewebluetoothService;
- LeweDatabaseService;
- LeweWebCloudService.

LeweService es el servicio principal de la aplicación que se inicia en el primer arranque y permanece en segundo plano hasta su cierre, pone en marcha los restantes servicios y se ocupa de finalizarlos una vez recibida la orden para apagar. El servicio se comunica directamente con los demás servicios LeweWebCloudService y LewebluetoothService; además de esto, envía las ordenes de conexión y desconexión a los servicios antes mencionados, que se comunican con el mundo exterior.

LewebluetoothService explota la clase BluetoothChatService para la recepción de datos y ha sido modificado para añadirle la posibilidad de recuperar la conexión con el dispositivo Bluetooth en el caso de que falle, y además se ha hecho compatible para operar con el protocolo Jack.

LeweDatabaseService se ocupa de gestionar la base de datos SQLite de los datos adquiridos.

LeweWebCloudService gestiona la comunicación entre la aplicación y la nube, en particular se ocupa de enviar los datos a la nube a través de HTML post y verificar la lectura correcta a través de la confirmación de recepción de los datos, tal como se prevé en el protocolo Jack. La nube permite el acceso en línea a sus datos desde cualquier lugar del mundo con la utilización de credenciales de acceso a tu propio espacio personal disponible de forma gratuita; si no dispones de una cuenta, hay un servicio de registro. Los datos se representan gráficamente y en forma de tabla clásica (Fig.13); también hay un módulo para hacer una búsqueda por fecha. Si no se utiliza ningún parámetro, se muestra

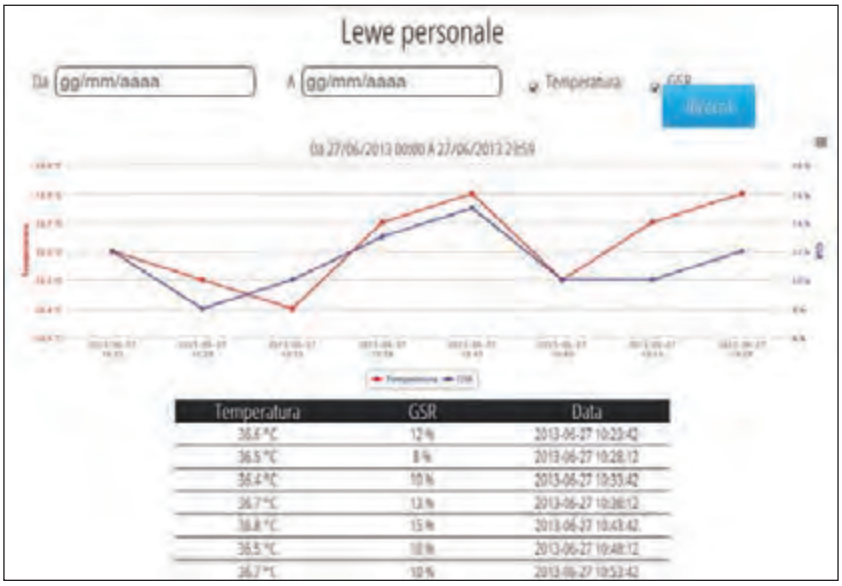

***Fig. 13** - Captura de la nube.*

automáticamente los datos relativos al último día de utilización de la pulsera. Mediante la configuración de la aplicación con los datos de acceso a tu espacio personal, puedes comenzar a cargar los datos biométricos recogidos por el brazalete. También es posible facilitar el acceso a sus datos a otras personas compilando con su username (nombre de usuario) el módulo presente en el página "Publico mi Lewe". Al acceder a la página "Lewe de un amigo" es posible, ver los nombres de los usuarios que nos han autorizado y, haciendo clic sobre uno de ellos, veremos los datos organizados de una manera similar a la página "Mi Lewe". La carga de los datos ha sido realizada por una sub-nube a la cual no se puede acceder directamente, sino que se lleva a cabo después de la autenticación efectuada por la aplicación Android para smartphone.

Tal sub-nube ha sido realizada con la ayuda del framework Azzurro y posee una implementación del protocolo de comunicación Jack para la recepción y la confirmación de los datos a almacenar.

Para activar el brazalete es suficiente alimentarlo mientras la aplicación Lewe.apk está instalada en un dispositivo Android.

En primer lugar es necesario vincular el brazalete al smartphone Accediendo a la pantalla de configuración, símbolo de engranaje en la esquina superior derecha (Fig.14). Desde el menú ajustes, es posible configurar la comunicación con el brazalete y la nube de internet (Fig. 15). Al hacer clic sobre el botón Lewe y sucesivamente sobre el botón ON, se inicia la búsqueda de dispositivos Lewe: en la casilla inferior aparecen los nombres de las pulseras que la aplicación detecta (Fig.16); gracias a un control del nombre de dispositivo Bluetooth, sólo se muestran los que responden al patron LEWE_idDevice, donde IdDevice es un número de identificación del dispositivo. Seleccionando la pulsera a la que se quiere conectar, se establece la conexión y la aplicación se ejecuta. Este paso sólo es necesario para el primer arranque de la aplicación, o cuando se cambia de dispositivo,

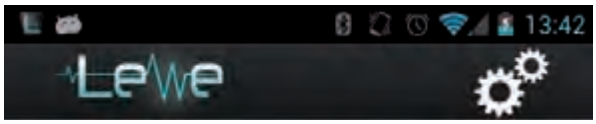

***Fig. 14** - Aplicación Lewe.*

***Fig. 15** - Configuración de la app.*

***Fig. 16** - Configuración de la app.*

***Fig. 17** - Configuración de la nube.*

en cuanto la aplicación memoriza el dispositivo Bluetooth al cual se conecta, intentará conectarse automáticamente cada vez que se abra la aplicación. Para la nube, es necesario hacer clic sobre "Web Cloud" y escribir la dirección de correo electrónico y contraseña en los campos adecuados (los que dan acceso a la nube web online) y marcar "Usar Web Cloud" (Fig. 17). La aplicación también puede funcionar sin la nube y por tanto su configuración es opcional. El sistema ha sido ampliamente probado con smartphones Samsung Galaxy Nexus, Galaxy Nexus 4 e Galaxy S2 con sistema operativo Android 4.0. Es posible ver la nube online accediendo a la página web http://www. lewe.tk. Para el acceso utilice el formulario haciendo clic sobre el botón "login", usando las credenciales: email prova@prova.it y contraseña Prova123.

CONCLUSIONES

No siempre es fácil desarrollar nuevas cosas, y mucho menos cuando pueden afectar a un gran número de personas y a una amplia gama de aplicaciones. Las nuevas tecnologías, especialmente los teléfonos móviles, ofrecen todavía un margen considerable para la innovación, más aún en los casos en que el software se encuentra con el hardware, y en esto Arduino nos ha enseñado mucho..

El MATERIAL

Todos los componentes utilizados en estos proyectos son fáciles de encontrar. El cargador solar de baterías de 1200 mAh cod. SOL20 cuesta 22,00 euros, el módulo Bluetooth cod. FT1032M cuesta 34,00 Euros, el Arduino Mega2560 REV3 (Cod. ARDUINO - MEGAREV3, sale por 51,00 Euros y el RTC cod. RTCSHIELDKIT por 11,00 Euros.

Los precios incluyen IVA.
Los gastos de envío no van incluidos.
Puede hacer su pedido en:
pedidos@nuevaelectronica.com