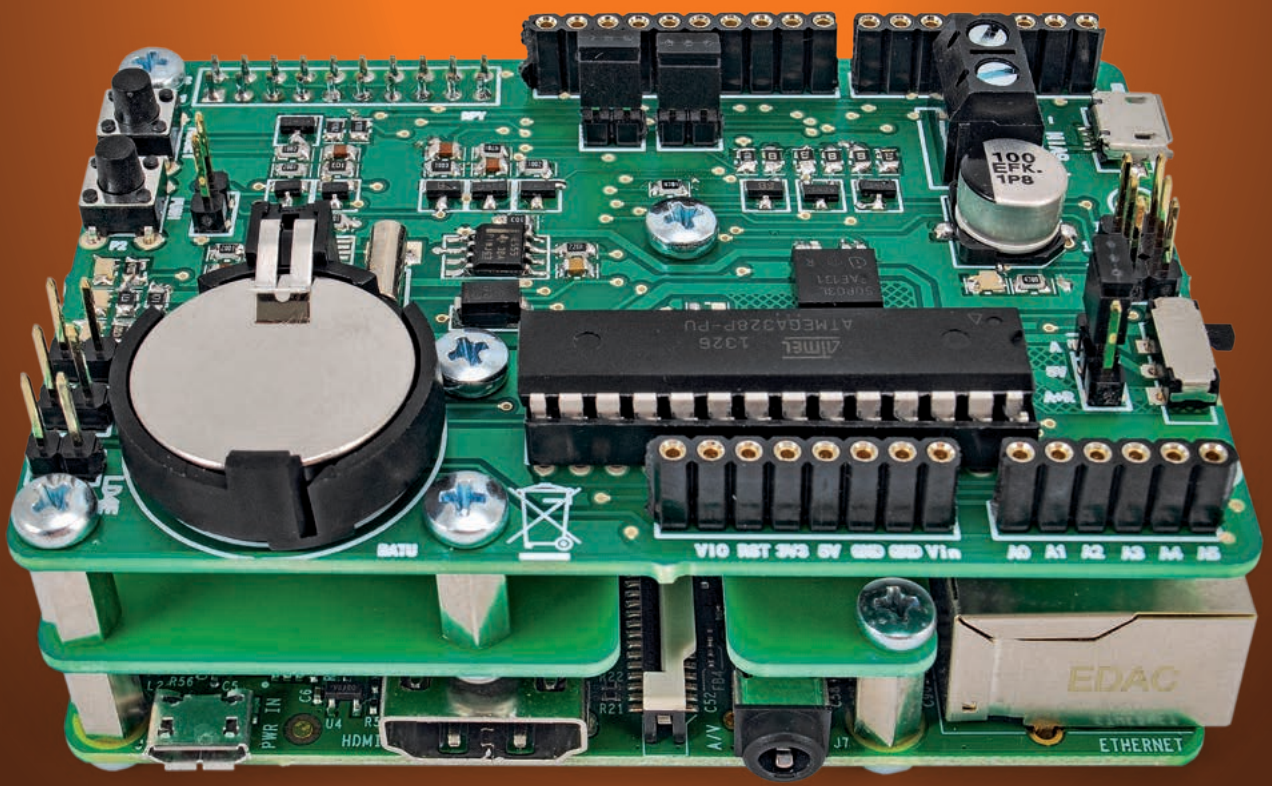
UTILIZAMOS RandA

DANIELE DENARO

Instalamos el software, comprobamos el funcionamiento de tarjeta y realizamos un Servidor Web, terminando con un ejercicio de programación.

Si os perdisteis los anteriores artículos, resumimos brevemente que es RandA: se trata de una tarjeta que permite la integración física y funcional entre Raspberry Pi y Arduino, permitiendo utilizar la amplia oferta de shields disponibles para Arduino combinando el enorme potencial de Raspberry Pi. Además RandA incluye una gestión inteligente de la alimentación y un reloj (en la **Fig. 1a** podéis ver el esquema funcional de RandA, mientras en la **Fig. 1b** encontráis los comandos y conexiones). Para controlar la tarjeta hemos previsto un set de software que se guarda en una SD-Card para insertar en el slot especifico de Raspberry Pi; por tanto, la primera cosa que hay que hacer después de haber montado RandA y haberla fusionado con Raspberry Pi, es instalar en la SD el software necesario para el funcionamiento del sistema. De hecho, sin eso el hardware del interruptor electrónico, del reloj RTC y el mismo Arduino no podría colaborar con Raspberry Pi. Además en el software está incluida también aquella parte a instalar en vuestro PC para



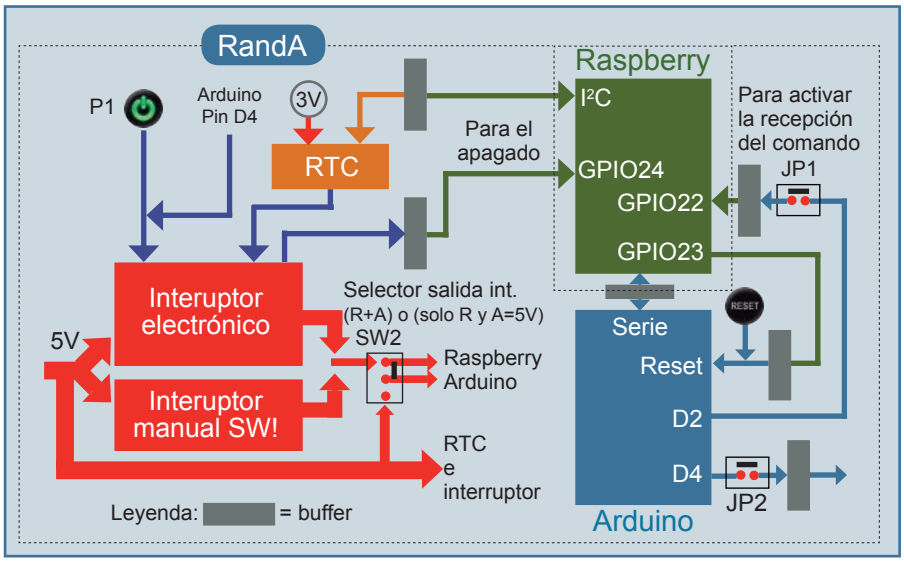
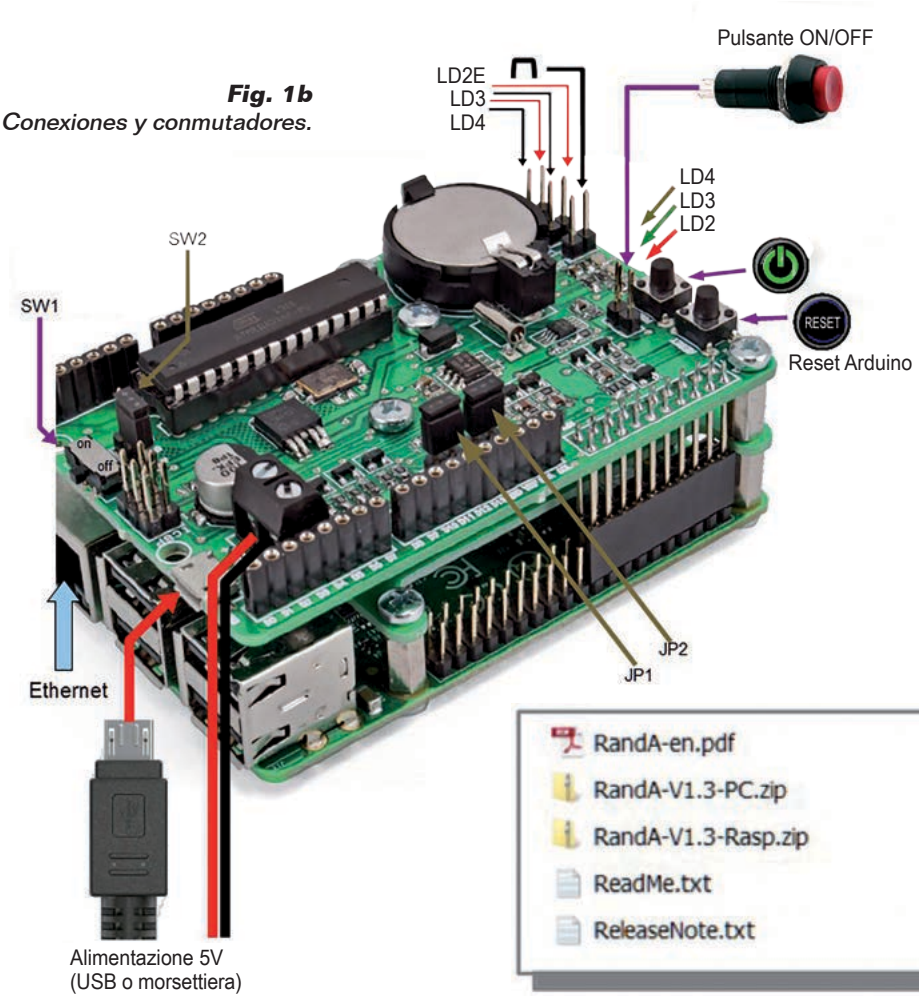


***Fig. 1a** - Esquema de bloques de RandA.*

Fig. 1b
Conexiones y conmutadores.



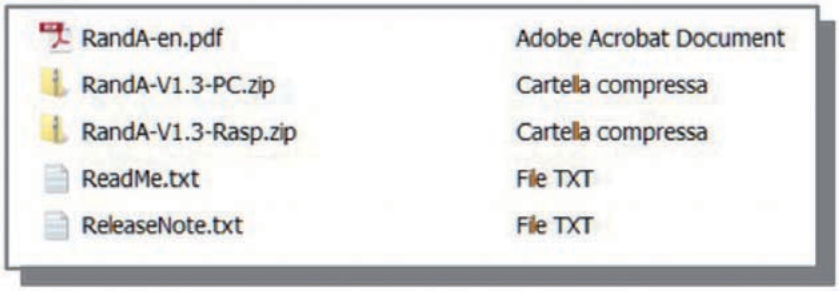


***Fig. 2** - Software de instalación "RandA-VI.3.zip".*

modificar el IDE de Arduino de manera que sea capaz de utilizar los puertos remotos y cargar los sketchs en la RandA conectada en red. En realidad esta parte debéis instalarla también si tenéis la SD-Card ya configuraba que se vende en nuestra tienda.
En este caso la carpeta con los archivos de instalación la encontráis en el directorio */home/pi/RandA*.
El software se puede descargar desde nuestra web junto a los archivos del proyecto; esta en formato de archivo comprimido (zip) y contiene a su vez dos archivos: uno para vuestro PC y uno para la Raspberry Pi sobre el cual está montada RandA.
Además contiene también un manual en Ingles (**Fig. 2**). El proceso a seguir para la instalación del software es el siguiente:

1. extraer el archivo para PC, y desde este extraer a su vez los archivos que irán colocados en la carpeta "lib" del IDE de Arduino (utilizar solo la versión 1.0.5);

2. antes, por seguridad, renombrar los tres archivos que sustituís. El IDE de Arduino estará ahora ya modificado; completar la operación extrayendo la librería RAComm de la carpeta "libraries" y colocándola en la carpeta "libraries" del IDE modificado;

3. extraer el archivo "RandAinstall.tar.gz" y el archivo "RandAinstall.sh" de la carpeta para Raspberry Pi.

En este punto debéis trabajar sobre Raspberry Pi. Ya que las instalaciones Raspbian (recordamos que el software está pensado para este entorno operativo) están configuradas con el server SSH ya activo, podemos conectarnos a Raspberry Pi vía LAN con un software que prevea este protocolo; en este artículo suponemos que utilizáis MobaXterm, que contiene también un X window server y una cómoda ventana FTP además de la consola Linux.
Por esta razón, identificada la dirección LAN de Arduino (podemos identificarlo a través del router o a través de las utilidades que hacen el scanning de la red como "Advanced Ip scanner"), abrimos una sesión SSH con ella (**Fig. 3**). Vemos enseguida que al lado de la consola esta la ventana FTP.
Solo nos queda hacer la descarga de los dos archivos sobre Raspberry Pi: los podemos meter en la carpeta */home/pi*. Antes de lanzar el script RandAinstall.sh es necesario hacerlo ejecutable con el comando:

```
"sudo chmod 777 RandAinstall.sh"
```

Ahora solo nos queda lanzar el script a través del comando (suponiendo que estamos en la direccion */home/pi*):

```
"./RandAinstall.sh"
```

¡Y eso es todo! Recordamos que como al final el script instala la Arduino IDE para Linux y "codeblocks" desde la web del software para Raspbian, debéis tener activa la conexión Internet. En el caso que no tuvierais cone-

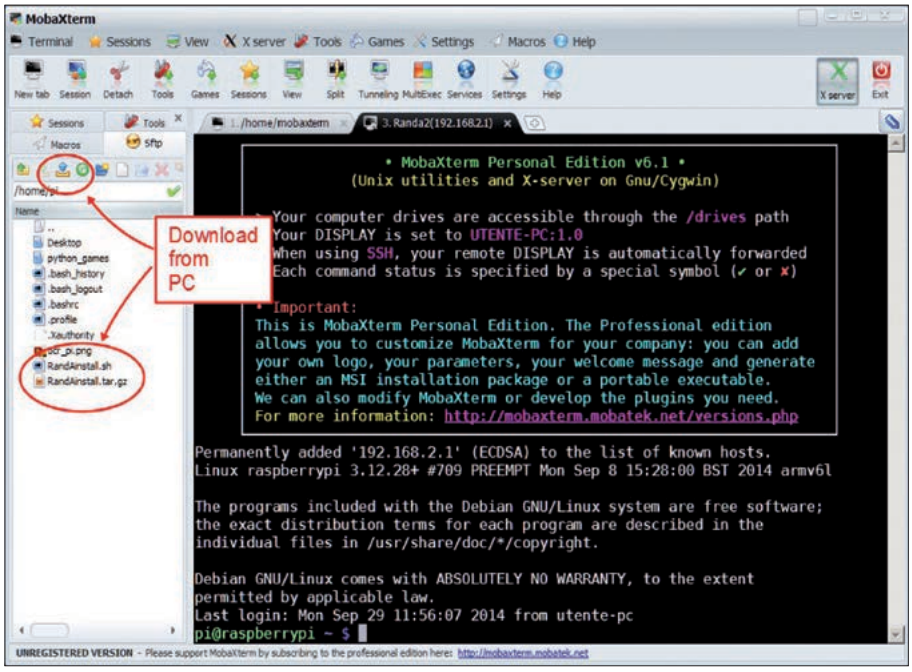


***Fig. 3** - Consola MobaXterm.*

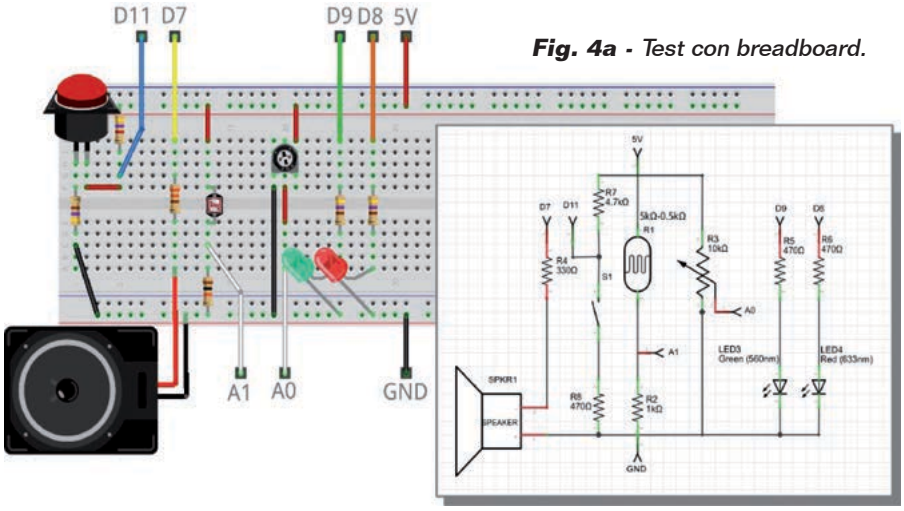


***Fig. 4a** - Test con breadboard.*

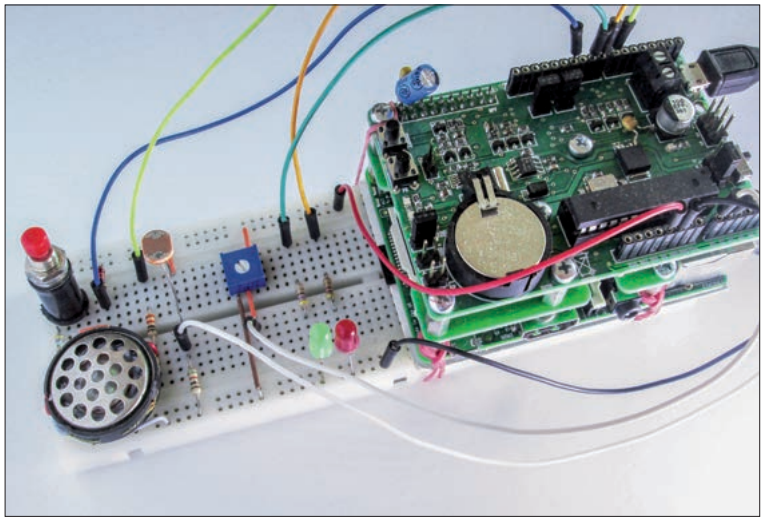


***Fig. 4b** Foto breadboard.*

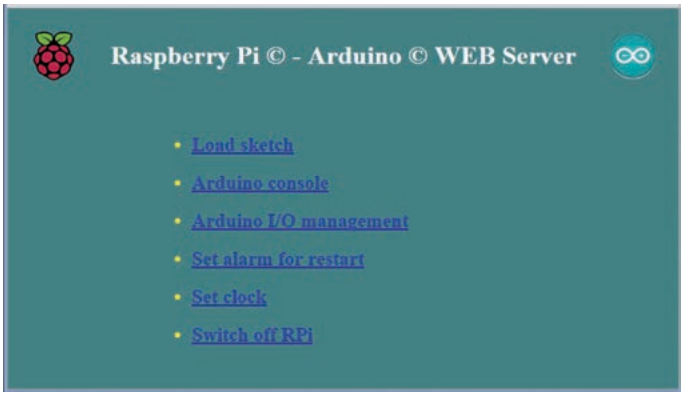


***Fig. 5** - Menú general de Arduino Servidor Web.*

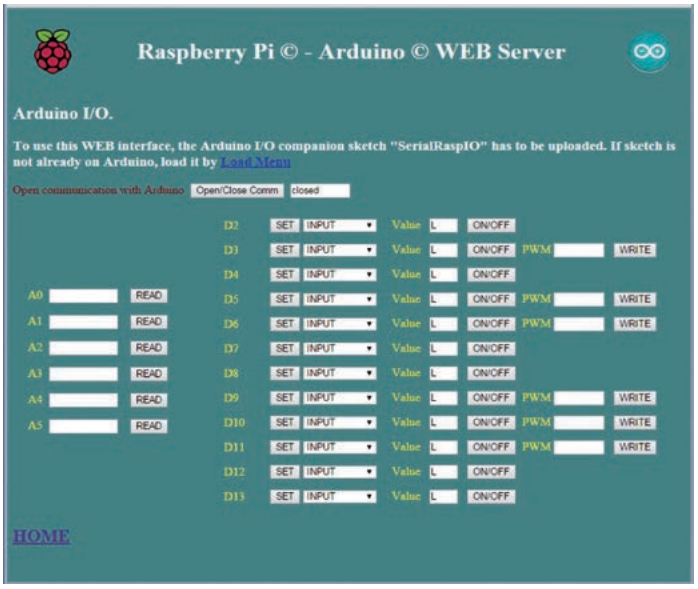


***Fig. 6** Pantalla I/O.*

xión a Internet, podéis instalar "codeblocks" y Arduino IDE en segundo plano, con los comandos:

```
sudo get-apt install codeblock
sudo get-apt install arduino
```

Pero después de la instalación de Arduino debéis modificar manualmente el IDE también sobre Raspberry Pi haciendo referencia a las tres líneas que encontráis en el script de instalación en la sección relativa a la modificación de Arduino IDE.

TEST DEL SISTEMA

Hagamos ahora pruebas para comprobar el funcionamiento de RandA y familiarizarnos con este sistema. Lo primero es reiniciar el sistema para hacer operativo el software; lo hacemos utilizando el comando:

```
"sudo reboot"
```

El sistema se apaga (la sesión SSH se cierra) y se reanuda. Ahora deberíais ver el LED amarillo apagarse después de un cierto tiempo: de hecho este permanece encendido durante el arranque y su apagado significa que el sistema está listo para trabajar. Abrimos de nuevo la sesión SSH y deberíamos ver también el nuevo mensaje de bienvenida, con el listado de los comandos específicos para RandA. Haciendo el "refresh" del recuadro *sftp* (botón derecho del ratón), veremos también las nuevas carpetas creadas en */home/pi*. Podemos probar la colaboración Raspberry-Arduino utilizando una placa de inserción rápida (breadboard) como en la **Fig. 4**. Usaremos el comando "ArduIO -h" si queremos obtener ayuda. Ahora podemos probar a encender el LED rojo ejecutando en secuencia los comandos:

```
ArduIO -set 8 out
ArduIO -wrd 8 1
```

Se no disponéis de una breadboard, podéis también testear el I/O de Arduino utilizando el LED standard incluido en la palca (pin 13). Para ello deberéis hacer: ArduIO -set 13 out y ArduIO -wrd 13 1. ArduIO utiliza el sketch SerialRasp. Si no lo encuentra en Arduino lo instala, lanzándolo de nuevo. El sketch utilizado se encuentre también en formato "fuente" en la subcarpeta "schetch4cmd" de "*/home/pi/bin*".
Para apagarlo es necesario ejecutar el comando:

```
ArduIO -wrd 8 0
```

De la misma manera podemos trabajar con el LED verde (pin 9); también este podríamos encenderlo con intensidad variable con el PWM (pero solo el 9); en este caso el comando es (para tener 50/255= 1/5 de la intensidad):

```
ArduIO -wra 9 50
```

Leemos el valor de la fotorresistencia en A1 (o del divisor de tensión en A0) con el comando:

```
ArduIO -rda 1
```

Mandamos una onda cuadrada a la frecuencia de 600 Hz en D7 con el comando:

```
ArduIO -pou 7 600
```

Para bloquear la onda, ejecutamos:

```
ArduIO -puo 7 0
```

Naturalmente la potencia del sonido es irrisoria debido al control directo de Arduino; en el caso quisiéramos tener un sonido más decidido, es necesario un buffer o una conexión a un amplificador. Finalmente podemos leer la duración de un impulso (del valor 1 al valor 0) en D11, con el comando:

***Fig. 7** - Load sketch.*

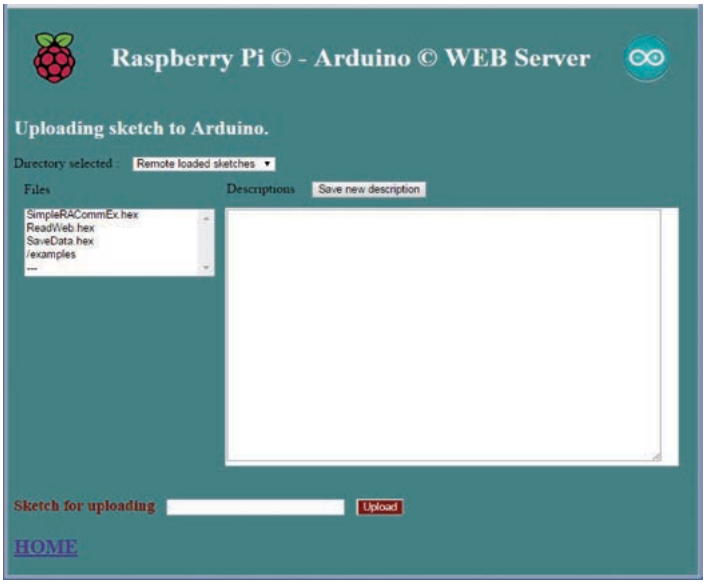
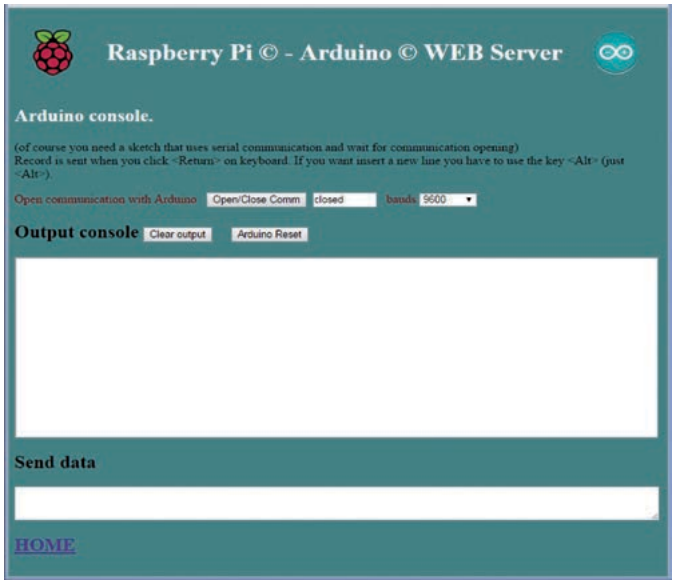


***Fig. 8** - Consola Arduino.*




```
ArduIO -pin 11 0
```

En el caso quisiéramos medir la duración del valor 0, deberíamos haber escrito:

```
ArduIO -pui 11 1
```

Si usáis el pulsador, recordar que el timeout es de un segundo. Finalmente podemos usar el pulsador como entrada digital:

```
ArduIO -rdd 11
```

En realidad podríais dialogar con Arduino utilizando directamente el puerto serie:

```
echo "WD13=1" > /dev/ttyS0
```

Este comando enciende el LED de Arduino (pin 13), si en Arduino está cargado el sketch "SerialRasp" (el utilizado por el comando ArduIO); de hecho corresponde al que hace el comando "ArduIO -wrd 13 1". El protocolo utilizado por SerialRasp es muy simple y esta descrito en el fuente del sketch.

SERVIDOR WEB

Probemos ahora el encendido a través del browser. Si trabajáis en LAN, insertar la dirección numérica de RandA, en vuestro browser preferido, y os aparecerá la página de inicio del server.
Si acabáis de encender RandA, tened un poco de paciencia antes de acceder a la dirección; esperar unos diez segundos después de que el LED amarillo del startup se haya apagado para dar tiempo a inicializarse al servidor web.
Hacer clic sobre "RPi&Arduino management" y os aparecerá el menú general de **Fig. 5**. Podéis probar a utilizar la misma breadboard para testear el dialogo a través de web: a propósito, haced sobre "Arduino IO management" y os aparecerá la pantalla de la **Fig. 6**.
Si habéis testeado ya la breadboard con los comandos ArduIO, sobre Arduino ha permanecido el sketch que gestiona los pines, que es sustancialmente el mismo utilizado por esta aplicación.
De otra manera debéis primero cargarlo utilizando la página "Load sketch"; en esta pantalla, la primera cosa que hay que hacer es abrir el dialogo a través del puerto serie (prestad atención hasta que no aparezca open).
Después podéis utilizar los distintos pulsadores para configurar y modificar o leer los valores: por ejemplo para leer el valor de iluminación detectado por la fotorresistencia, haced clic en el botón <READ> correspondiente a A1.
Volvamos al menú y pasemos a la página "Load sketch" (**Fig. 7**): en ella podéis ver que tiene dos recuadros principales, uno con un listado de sketch y uno que contiene le descripción que vosotros podéis insertar o modificar directamente; en realidad los listados son dos, conmutables a través de la selección de la lista de desplazamiento. De hecho una carpeta contiene los sketch creados con el IDE remoto y la otra aquellos creados con el IDE local
(sobre Raspberry Pi).
Seleccionamos el sketch "TestSerial.hex" dentro de la carpeta "/examples" : nos aparecerá la descripción. Ahora clicamos sobre la tecla <Upload>: debería aparecer el mensaje de uploading realizado con éxito (si alguna cosa no va por el sentido correcto, haced un reset o haced de nuevo un upload). Ahora solo nos queda utilizarlo.
Ahora volvemos al menú y seleccionamos la opción "Arduino console" (**Fig. 8**); lo primero

comprobamos que la velocidad seleccionada sea 9.600 baud y después abrimos el puerto serie. Esperamos hasta que no aparezca "open" y hacemos clic sobre el pulsador <Arduino Reset> para volver a arrancar el sketch. Ahora podemos modificar el tiempo de parpadeo insertando un valor en milisegundos (>100) en el recuadro "Send data" y pulsando envió. Acordaos de cerrar el puerto serie cuando abandonéis la página.

Después podremos ir a la página "Set clock" e insertar el valor correcto para el reloj. Lo podemos tomar automáticamente del de sistema; de hecho si Raspberry Pi está conectada a Internet debería tener la fecha actualizada. Ahora podremos utilizar la página "Set alarm for restart" para fijar un horario de reencendido automático y, por tanto, apagar RandA a través de la página "Switch off RPi" que lanza un shutdown y cierra la alimentación.

RANDA SE CONVIERTE EN UN ARDUINO MEJORADO

Supongamos que hayáis adquirido también la tarjeta SD ya configurada: entonces la instalación se reduce a la modificación del IDE Arduino que ya utilizáis en vuestro PC. Esta operación ha sido descrita al inicio del artículo. El archivo de instalación lo encontráis como backup en la carpeta */home/pi/RandA*; solo debéis copiaros el archivo sobre el PC, por ejemplo utilizando la ventana FTP de la sesión MobaXterm, y proceder como ya se ha comentado, pero solo para la parte que comprende el software para PC.

Una vez sustituidos los archivos, lanzar el IDE de Arduino sobre vuestro PC. Podrá pareceros que tardáis un poco más que antes porque tiene que buscar la red para encontrar puertos remotos RandA. Apenas abierto podéis comprobar que entre los puertos disponibles aparece también uno remoto, que contiene una dirección de red, por ejemplo "//192.168.1.8/Arduino": seleccionarla.

Para hacer una prueba podéis utilizar el clásico Blink o un sketch vuestro. Tened presente que cualquier sketch que carguéis sobre RandA, seguirá estando en copia en la carpeta dedicada al desarrollo con IDE remoto. Los sketch estarán sin embargo solo en formato ejecutable (.hex), pero tendrán el mismo nombre del fuente.

Probamos a utilizar el sketch "TestSerial" que hemos utilizado ya con el Servidor Web. El código fuente lo encontráis en la carpeta "examples-notRAComm" de la librería RAComm. En esta carpeta hemos puesto ejemplos para utilizar con RandA que no hacen uso propiamente de la librería.

Si no ha sido cargado anteriormente, cargamos el sketch a través del puerto remoto. Ahora, para utilizarlo e interactuar a través del puerto serie, abrimos simplemente la consola del IDE: justo como haríamos si tuviéramos Arduino conectado por USB. Conviene también hacer un reset, porque hay que tener presente que ahora no está el automatismo del reset a la apertura de la consola. El reset podéis hacerlo manualmente, o por remoto utilizando el comando Linux "ResetRandA", o incluso a través del Servidor Web desde la pági-

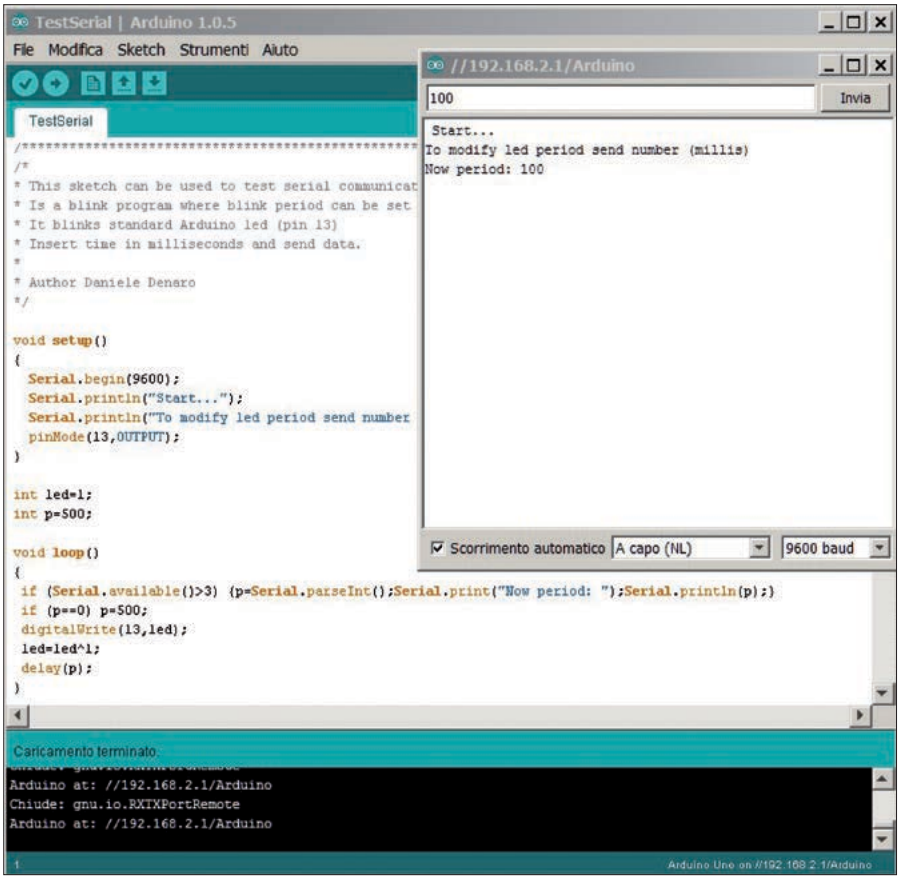


***Fig. 9** - El sketch TestSerial del IDE Arduino.*

na "Arduino console" (**Fig. 9**); en este caso no es necesario abrir la conexión serie.
Probamos a utilizar un ejemplo con el uso de la librería que dialoga con Raspberry Pi. El sketch más simple contenido en los ejemplos de la librería es "BasicRACommExample". Antes de cargarlo conviene abrir una sesión MobaXterm (si no está abierta ya) porque cuando se usa la librería la consola del IDE no se puede utilizar ya que el puerto serie está gestionado por el programa Linux que dialoga con Arduino para ejecutar sus comandos. Por lo tanto para visualizar los contenidos es necesario escribirlos sobre un Xterminal abierto para la ocasión (**Fig. 10**).
Cargado el sketch, este debería abrir una ventana Xterminal y visualizar el timestamp pedido a Raspberry Pi. Después de unos segundos la ventana se cierra y el sketch hace parpadear el LED tantas veces como sea la hora actual. La lectura del reloj se produce a cada reset. El reset puede hacerse manualmente o por vía remota a través del comando "ResetRandA" o a través Servidor Web.
En la carpeta de los ejemplos están contenidos otros sketch más complejos: uno de ellos guarda en el archivo algunos datos de sensores analógicos y digitales cada hora, mientras otro descarga desde una web datos meteorológicos y los guarda en un archivo.
Finalmente en la carpeta de ejemplos hay un sketch que utiliza la modalidad "Arduino siempre encendido". En este sketch, el valor sobre un pin analógico decide si encenderá Raspberry Pi -si no estuviese ya encendido- para después escribir sobre un archivo de log el evento y apagar de nuevo Raspberry Pi en el caso que lo hubiese encontrado apagado. Con este sketch es necesario por tanto poner el conmutador SW2 en la modalidad "Arduino siempre encendido" y tener el puente sobre JP2 (que conecta el interruptor con el pin D4 de Arduino).
Para el experimento podéis utilizar la breadbord con la fotorresistencia como se ilustra al inicio del artículo: veréis que, oscureciéndola, Raspberry Pi se enciende (si no lo está ya) y arranca el registro del evento.

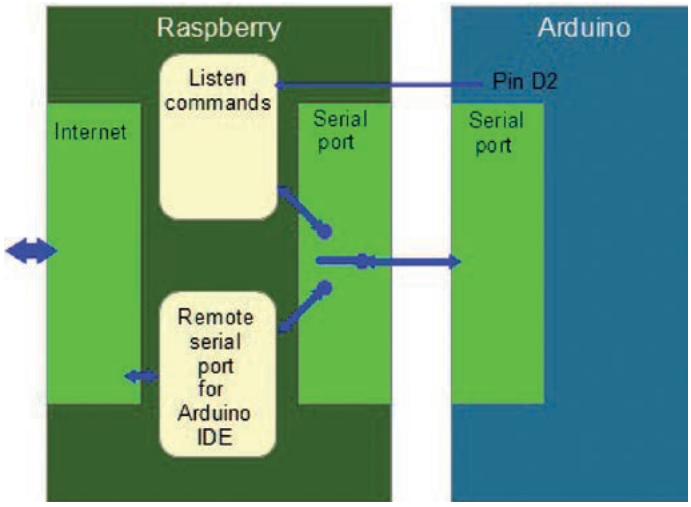


Fig. 10 *Puerto serie al servicio de distintos programas.*

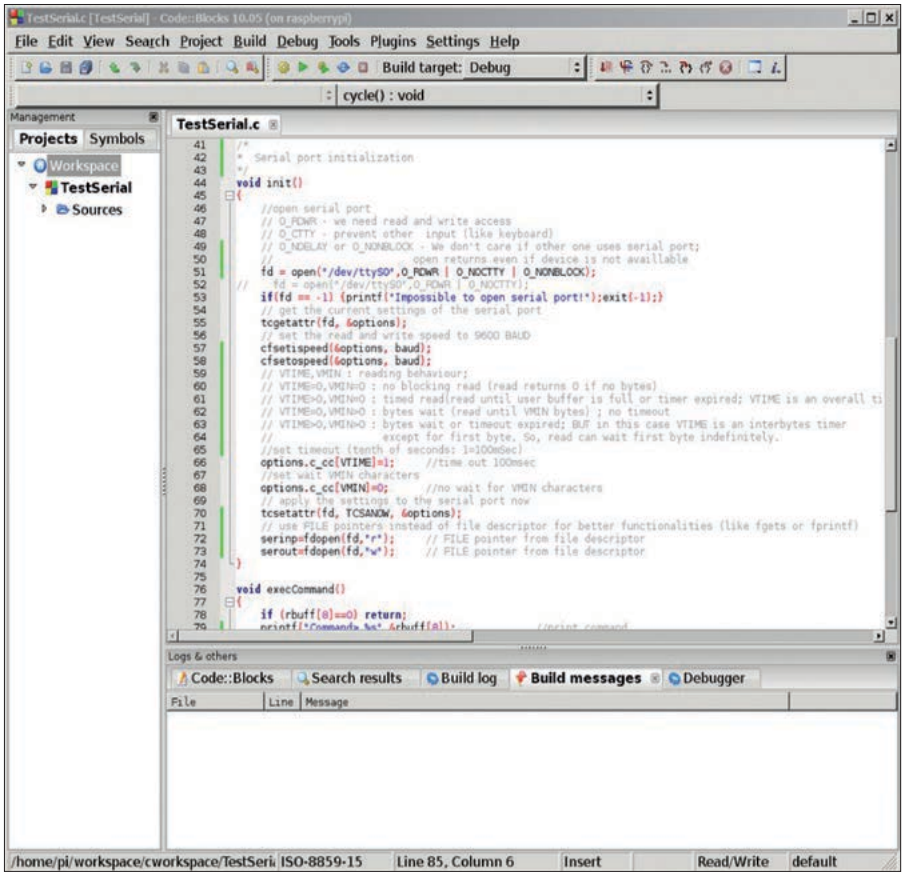


Fig. 11 *- Codeblocks.*

el MATERIAL

Este proyecto puede ser fácilmente realizado por cualquiera que tenga un mínimo de experiencia en el montaje manual de componentes SMD y disponga de las herramientas necesarias. La placa RandA esta también disponible ya montada y probada (incluyendo las piezas pequeñas) al precio de 39,00 Euros IVA incluido. Está disponible también la board Raspberry PI Tip B+ (cod. 8111284RS) al precio de 36,00 Euros

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

PROGRAMAMOS RANDA

Si en el ámbito Arduino la programación se hace a través del ampliamente conocido entorno IDE, en el ámbito Raspberry Pi la elección es más amplia: mientras tanto podéis utilizar los comandos ya preparados y catalogados en /home/pi/bin. Recordamos, en todo caso, que para moveros en el "file system" de Raspberry Pi, podéis utilizar el recuadro "Sftp" de la sesión MobaXterm, o, mejor aún, lanzar la interfaz gráfica del archivo manager "pcmanfm" que nosotros hemos renombrado "explorer".
Mas en general, para programar, podéis utilizar el lenguaje de script que mejor conocéis o un lenguaje completo de programación como C++ (o Java). Encontrareis dos ejemplos de script en /*home/pi/bin/examples*: uno de ellos hace uso de bash y el otro de Python. Probemos ahora a hacer un programita en C y con ese objetivo lanzamos "codeblocks". Tener presente que el arranque podría ser un poco largo y sobretodo sin señales de "work in progress"; además estar atentos a que si alguna ventana que espera una validación hubiera quedado bajo las otras, codeblocks podría no reaccionar. En el primer arranque, codeblocks pide el compilador a utilizar entre los encontrados: seleccionar el primero, que es Gnu GCC compiler, como compilador predefinido. Una vez abierto codeblocks podéis buscar el proyecto de ejemplo más elemental que se encuentra en */home/pi/workspace/cworkspace/TestSerial/TestSerial.cbp*. En codeblocks, los proyectos se abren haciendo referencia a los archivos ".cbp". La ventana codeblocks está constituida por distintos sectores, los principales son el del listado de los proyectos abiertos con su estructura y el del editor de las fuentes. La salida sobre la consola se muestra sin embargo abriendo automáticamente una ventana Xterminal. El programa TestSerial.c (no confundir con el sketch Arduino TestSerial visto anteriormente) es el fuente catalogado en el proyecto y es un ejemplo de cómo se abre un puerto serie para comunicar con Arduino. El programa abre el puerto serie, lee los registros que Arduino envía y los visualiza sobre la consola Linux; para visualizarlo, basta seleccionarlo en el recuadro del proyecto con un doble clic. Para utilizar el programa es necesario tener un sketch que envié registros a Raspberry Pi.
Naturalmente codeblocks, como todos los entornos de programación permite proceder en modo "debug" (depuración) paso a paso o definiendo los "break-point". A través de la voz "Debugging windows" es posible abrir una ventana para "watches" (lectura de los contenidos de las variables) y otros. Los comentarios presentes permiten describir la configuración básica del puerto serie que permiten poner las bases para un programa de colaboración y uso de Arduino. En la web de Internet de codeblocks (www.codeblocks.org) encontrareis el manual y otras informaciones. En la carpeta "/home/pi/workspace/cworkspace" están contenidos también todos los programas que tienen su versión ejecutable en "/home/pi/bin" (pero también aquellos utilizados en /etc). Si después queréis integrar el Servidor Web con vuestras páginas, lo podéis poner en "/home/apache-tomcat-7.0.47/webapps/ROOT". Una web-application (file .war) la podéis descargar a través del "manager" de Tomcat (username y password: tomcat). Finalmente si queréis utilizar el camino de los script CGI, podéis colocarlos en "/home/apache-tomcat-7.0.47/webapps/ROOT/WEB-INF/cgi", pero recordar referenciarlo como "http://...../cgi-bin/....". En la carpeta podéis encontrar ya un par de script de ejemplo.

(192063)