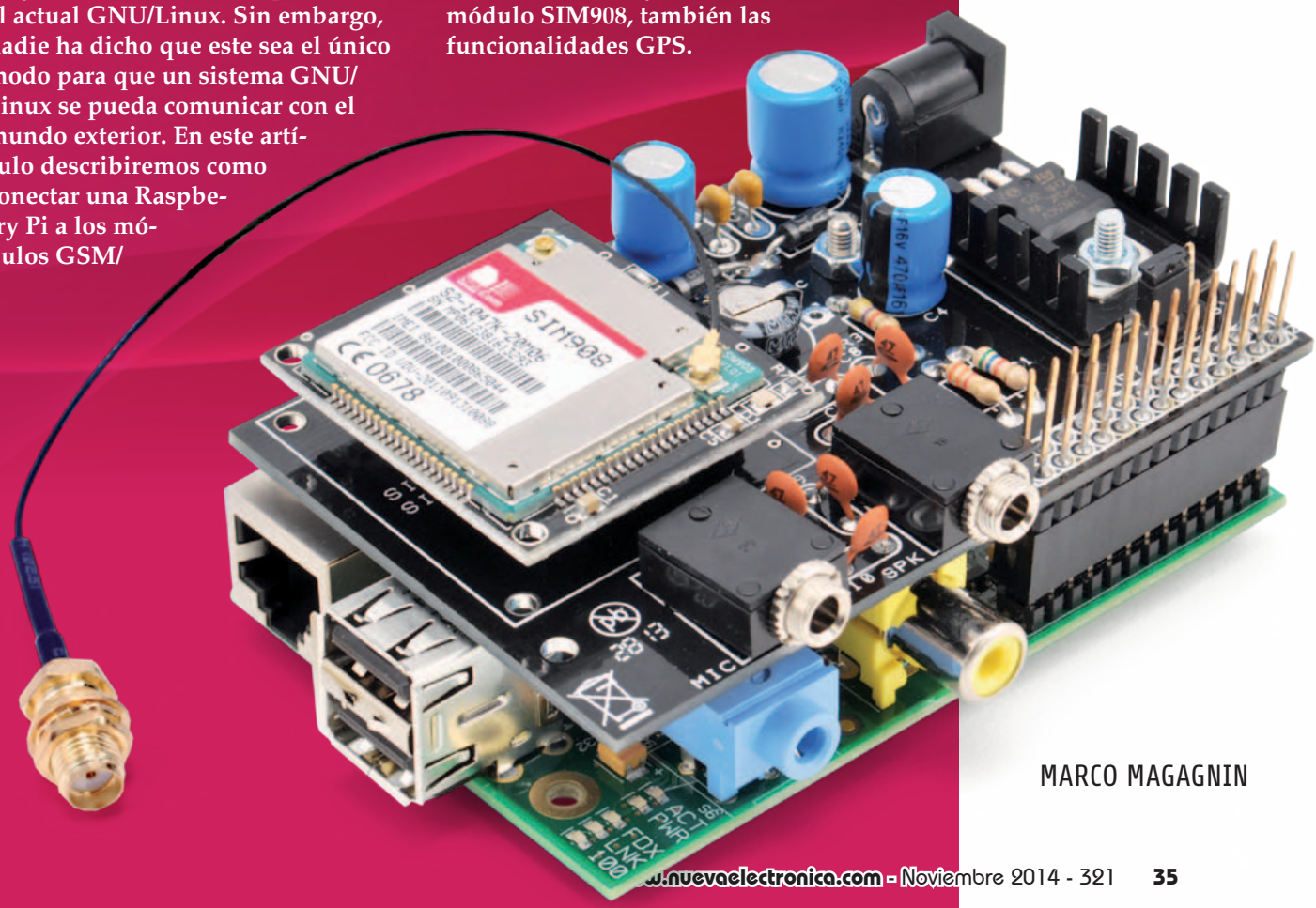
SHIELD DE ESPANSIÓN GSM/GPRS/GPS PARA RASPBERRY PI

Shield de expansión para Raspberry Pi con el que, añadiendo el módulo GSM/GPRS SIM900 o SIM908 (con GPS), podrás realizar aplicaciones que pueden controlarse y comunicarse a través de la red de telefonía móvil.

El sistema operativo GNU/Linux nació con la "vocación" de la conectividad a través de Internet. Los protocolos TCP/IP (Transport Control Protocol – Internet Protocol) están ligados a la historia de UNIX primero, y de sus sucesores después, hasta el actual GNU/Linux. Sin embargo, nadie ha dicho que este sea el único modo para que un sistema GNU/Linux se pueda comunicar con el mundo exterior. En este artículo describiremos como conectar una Raspberry Pi a los módulos GSM/GPRS de la familia SIM9XX, pero es igualmente válido para todos los sistemas GNU/Linux dotados de puerto serie. De este modo es posible extender las funcionalidades de la Raspberry Pi para aplicaciones que necesiten conectividad móvil y, utilizando el módulo SIM908, también las funcionalidades GPS.



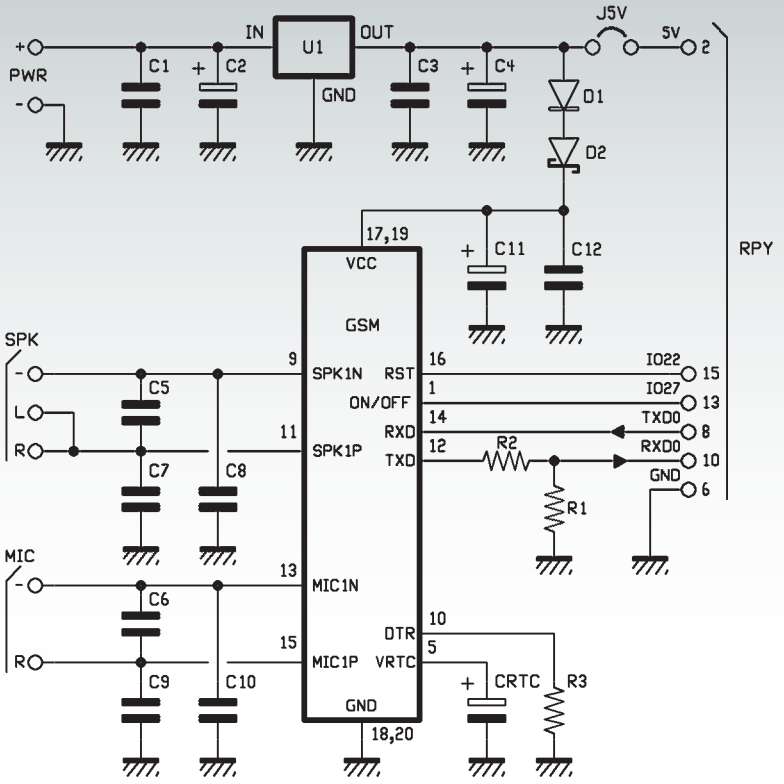
MARCO MAGAGNIN

No debemos pensar en la conectividad GSM sólo para aplicaciones estrictamente "móviles" como aquellas sobre vehículos o en lugares no conectados a la red internet. Pensemos simplemente en un sistema domótico, antirrobo o de control de una instalación desatendida.
En la mayoría de las aplicaciones si falla el suministro de energía eléctrica, también fallaría la conexión internet. En ese caso, la única posibilidad de enviar un mensaje de aviso del estado de las cosas es mediante una conexión a la red de telefonía móvil. Para una máxima fiabilidad el sistema de control debería estar dotado de una batería de respaldo para, si se produce un corte en el suministro eléctrico, poder comunicarse con el sistema GNU/Linux.
Estamos pensando en ello. Por ahora retornemos a nuestro shield GSM/GPS y a cómo utilizarlo con la Raspberry Pi. Los shield dedicados a la Raspberry Pi son proyectados de forma que puedan ser apilados uno sobre el otro y poder así gestionar más funciones simultáneamente, de manera que se puedan integrar diferentes "servicios" en una única solución integrada. Podría ser el caso de un sistema domótico, dotado de sensores y actuadores, que utilice el LCD para la activación y desactivación, pero que haga posible la misma operación también de forma remota, bien vía Internet, bien por GSM.
Además de esto pueden añadirse servicios de asistencia para ancianos, activables tanto en local como en remoto, con funcionalidades vocales o de envío SMS a números predefinidos.
Podemos añadir la toma de imágenes con una cámara de vídeo, asociada a la apertura de las puertas y con reconocimiento facial o de movimiento, con el envío de mensajes y de las mismas imágenes en caso de ...sospecha.
El único límite es la fantasía y ... la capacidad de proyectar y realizar las aplicaciones. Aplicaciones de esta magnitud superan las posibilidades de los microcontro-

Módulos GSM/GPRS que utilizamos

El shield para Raspberry Pi presentado en este artículo puede albergar tanto el módulo SIM900 (GSM/GPRS) como el módulo SIM908 (+GPS).
Más precisamente el módulo SIM900 es un Quad-band compatible GSM/GPRS, operativo sobre frecuencias de 850/900/1800/1900 MHz, con SMS y accesos a Internet mediante stack TCP/IP integrado, gestionado por un procesador AMR926EJ-S que se ocupa también de la gestión de la comunicación telefónica mediante comandos AT. El módulo, compatible GSM fase 2/2+, es de clase 4 (2 W) a 850/ 900 MHz y de clase 1 (1 W) a 1800/1900 MHz; SIM900 integra una interfaz analógica, un A/D converter, un RTC, un bus SPI, un I²C, y un módulo PWM.
El módulo se alimenta con tensión continua con un valor comprendido entre 3,4 y 4,5 V, absorbe un máximo de 0,3 A (continuados) en transmisión. Está dotado de conector para antena externa.
El SIM908 integra un receptor GPS (con antena separada) de 42 canales capaz de una sensibilidad de -160 dB en tracking y -143 dB en adquisición, de una precisión en la localización de solo 2,5 m y de tiempos de operación de 1 s para el Hot Start y 30 segundos para el Cold Start. El set de comandos AT se extiende por el acceso al GPS.
Los manuales están disponibles en PDF en internet buscando "SIM900 AT command" y "SIM908 AT command".

ladores en general, aunque éstos se pueden utilizar de manera rentable en la gestión de los sensores y los dispositivos periféricos. Volviendo al tema concreto, claramente los distintos shield no deben utilizar los mismos pines, alimentación aparte, de manera que se eviten conflictos no gestionables vía software.
En el shield GSM/GPRS/GPS se utilizan los pines del conector GPIO de la Raspberry Pi conectados al puerto serie y otros dos pines de I/O para el encendido/apagado y el reseteo del módulo SIM9XX.

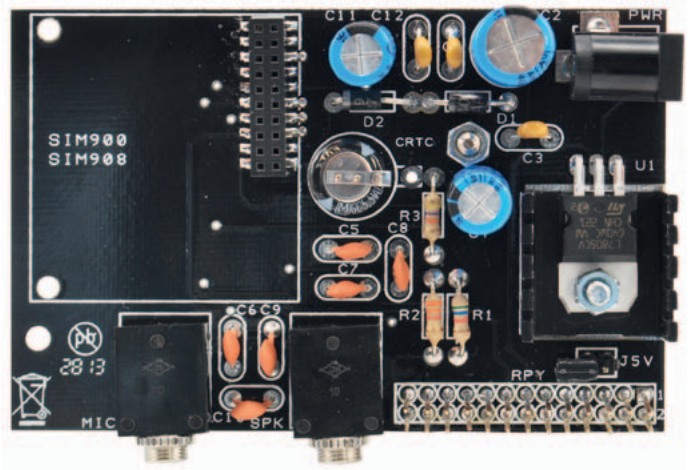


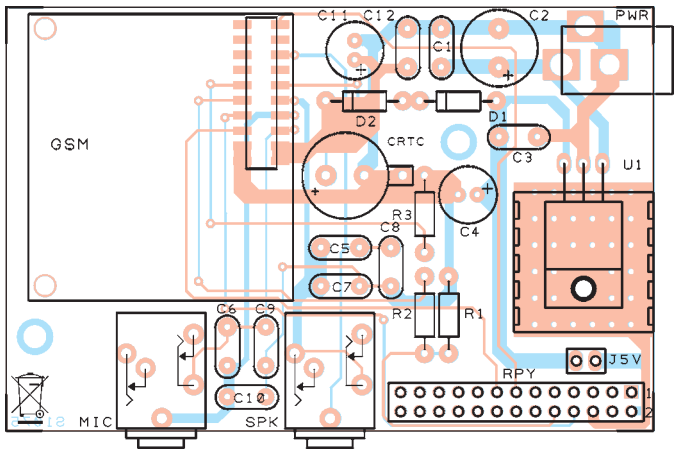
ESQUEMA ELÉCTRICO

El esquema eléctrico del shield FT1075 está diseñado para dar alojamiento físico para los módulos SIM900 y SIM908, proporcionar una alimentación autónoma al módulo GSM, exportar la entrada micrófono y la salida audio sobre dos jack dedicados y conectar correctamente los pines del conector GPIO de la Raspberry Pi con aquellos de los módulos GSM. Comenzamos por el circuito de alimentación compuesto por el regulador U1, un integrado 7805, y por la red de condensadores de filtrado y estabilización de la tensión (C1, C2, C3 y C4).
A través del diodo D1 y el zener D2 se lleva la tensión positiva a los pines VCC de los módulos SIM9XX, filtrada y estabilizada por los condensadores C11 y C12. Mediante el puente J5V es posible alimentar también la Raspberry Pi desde la misma fuente del shield. Al conector PWR se puede conectar un alimentador que entregue una tensión continua entre los 7 y los 12 Voltios. El pin ON/OFF del SIM9XX está conectado al pin 13 del GPIO de la Raspberry Pi (GPIO27) para el encendido y apagado del módulo por programa. El pin de reset, RST, del módulo SIM9XX está conectado al pin 15 de la Raspberry PI (GPIO22), de forma que puede ser reseteado por programa.
La conexión serie está disponible en los pines 8 (TXD) y 10 (RXD) de la Raspberry Pi. Sobre la línea de entrada RDX la adaptación al nivel de tensión de 3,3 V requeridos por Raspberry Pi se realiza por el divisor de tensión compuesto por R1 y R2. No es necesaria ninguna adaptación de tensión en el sentido opuesto ya que los 3,3 V de salida de la Raspberry Pi son compatibles con la entrada de los SIM9XX. El DTR se mantenido constantemente a nivel bajo mediante la resistencia R3. El pin para la alimentación del Real-Time Clock (VRTC) está conectado al condensador CRTC. Finalmente las líneas de entrada de micrófono y de salida audio son conectadas a los respectivos conectores MIC y SPK cada uno con su red de condensadores de filtrado, C6, C9 y C10 para MIC y C5, C7 y C8 para SPK.
En el diagrama de montaje es posible ver la disposición de los componentes. La única sugerencia, aparte las atenciones habituales, es posicionar correctamente el doble conector pinheader de forma que se distancie el circuito impreso del shield de la Raspberry Pi para permitir el montaje "en paquete", en el caso que se quiera utilizar conjuntamente con otros shield, como el shield

Fig. 1

[plano de **MONTAJE**]





Lista de materiales:

R1: 5,6 kohm
R2: 3,3 kohm
R3: 4,7 kohm
C1: 100 nF multicapa
C2: 470 µF 25 VL electrolítico
C3: 100 nF multicapa
C4: 470 µF 16 VL electrolítico
C5÷C9: 47 pF cerámico
C10: 47 nF cerámico
C11: 470 µF 16 VL electrolítico
C12: 100 nF multicapa
U1: 7805
D1: 1N4007
D2: 1N5819
CRTC: Condensador de perfil bajo 0,1F

Varios:
- Conector de alimentación
- Toma jack estéreo 3,5 mm para PCB (2 pz.)
- Disipador
- Tornillo 8 mm 3 MA
- Tornillo 12 mm 3 MA
- Tuerca 3 MA (2 pz.)
- Jumper
- Tira de 2 pines macho 2
- Toma MMS SMT 2x10 vías paso 2 mm
- Conector 13x2 para RaspberryPi
- Torreta M/H 18 mm
- Circuito Impreso

LCD que os hemos mencionado antes. Por tanto, antes de soldar los pines del conector conviene posicionar el conector de 26 pines como se describe a continuación:

- Posicionar el espaciador hexagonal, insertando el extremo roscado desde el lado de la soldadura en el agujero cerca del condensador CTRC y fijarlo con al tuerca M3;
- Posicionar el conector de 26 pines en la posición identificada con RPY, teniendo el lado con los terminales hembra hacia abajo;
- Poner el shield sobre la Raspberry Pi apoyando sobre este último el separador hexagonal. Unir los dos conectores de 26 pines, y fijar el separador con el tornillo M3;
- Una vez que las tarjetas están alineadas y separadas correctamente, podemos soldar los terminales macho del conector sobre el shield.

Siempre para garantizar la posibilidad de "empaquetamiento" colocar con cuidado el radiador del integrado U1, de manera que no toque el shield montado encima.

USO PRÁCTICO DEL SHIELD

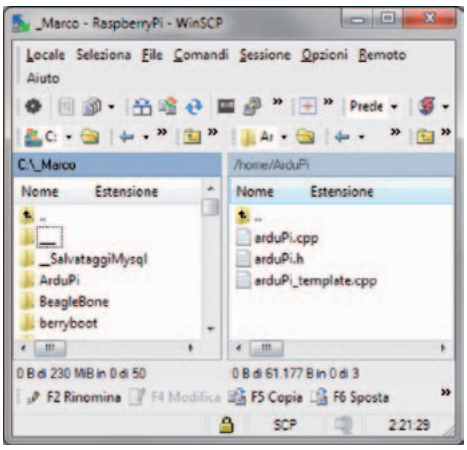
Como primera operación montamos sobre el shield uno de los módulos GSM compatibles y después, si aún no se ha hecho, montamos el shield sobre el conector de Raspberry Pi, poniendo atención que la parte inferior del shield no quede en contacto con los conectores USB o Ethernet. En caso de duda, es mejor proteger los conectores mismos con cinta aislante. Conectamos los periféricos, la red y la alimentación a la Raspberry Pi en el mismo modo.

Conectamos la alimentación al shield FT1075 y ... aplicamos tensión.

En este artículo os proponemos un enfoque didáctico del uso del shield GSM de tal manera que podamos experimentar el máximo antes de empezar a escribir aplicaciones más o menos complejas. Recordemos que usando GNU/Linux podemos realizar aplicaciones que desarrollan muchas funciones, compartiendo recursos como el shield GSM.

Por ejemplo podemos escribir una aplicación que permanezca a la espera de eventos externos, como la llegada de una llamada de voz o un mensaje SMS, pero que sea también capaz de reaccionar a eventos internos como la activación de entradas de I/O, tanto digitales como analógicas, o a la solicitud de ejecución de llamadas de voz, envío de SMS,

Fig. 2



individual o en masa, envío de archivos, mensajes de e-mail, etc. Al tiempo, se puede querer adquirir datos desde la sección GPS del módulo. Para hacer esto es necesario proyectar una arquitectura modular capaz de compartir los recursos sin crear condiciones de conflicto. El resultado es una arquitectura compuesta de módulos servidor que hacen de interfaz y controlan los recursos y los módulos operativos, cada uno especializado en una función específica que pueden ser llamados y ejecutados al verificarse condiciones específicas y requerir los servicios de comunicación de los módulos de servidor, que procederán a satisfacerlos según una lógica que garantice las prioridades y evite los conflictos. Para realizar esto es necesario profundizar en detalle las exigencias funcionales de la aplicación, y es el motivo por el cual, antes de adentrarnos en próximos números en el diseño de aplicaciones cliente/servidor, os proponemos un par de modalidades que permiten experimentar de manera rápida el uso del shield.
La primera modalidad permite reutilizar con pocas modificaciones los programas escritos para Arduino, con el fin de familiarizarse con el shield por herencia del mundo Arduino.
Para esta modalidad utilizaremos la librería arduPi realizada por Libelium Comunicaciones Distribuidas S.L., puesta a disposición en la licencia GNU General Public License y disponible desde CookingHacks.
La segunda es una modalidad directa. Nos conectamos al puerto serie con un programa TTY y controlamos el módulo GSM dando y recibiendo directamente los comandos de gestión. Esto nos permite probar las secuencias de comandos necesarios para realizar una funcionalidad específica sin la necesidad de escribir y testar cada vez un programa.

ENVIAMOS UN SMS AL ASTILO ARDUINO

Para instalar la librería arduPi es suficiente descargarla de la dirección:

http://www.cooking-hacks.com/skin/frontend/default/cooking/images/catalog/documentation/raspberry_arduino_shield/arduPi_1-5.tar.gz

La descomprimimos en la carpeta /home y, por comodidad, modificamos el nombre a "ArduPi". Siempre por comodidad meteremos en la misma carpeta los programas que vayamos a realizar. Antes de compilar la librería abrimos el programa arduPi.cpp y, si es distinta, sustituimos en la línea 64 la referencia al puerto serie existente con la línea:

```
serialPort="/dev/ttyAMA0";
```

Ahora vayamos a la carpeta ArduPi (**Fig. 2**) con el comando

```
cd /home/ArduPi
```

y compilamos la librería con el comando (**Fig. 3**):

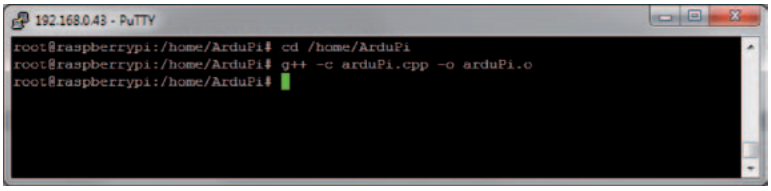
```
g++ -c arduPi.cpp -o arduPi.o
```

Con respecto a los programas os proponemos tres módulos. Dos permiten encender y apagar el módulo GSM y nos serán muy útiles en el futuro.
El tercero nos permite enviar un SMS a un teléfono móvil. Para encender el módulo es necesario poner a nivel alto el pin GPIO27 durante dos segundos y después llevarlo de nuevo a nivel bajo, si este está encendido será la misma operación para apagarlo. Para el reset es necesario poner a nivel alto el pin GPIO22 y mantenerlo así durante medio segundo para después volverlo a llevar a nivel bajo. Los programas se pueden descargar gratuitamente de la web *www.nuevalectronica.com*.
Primero de todo, para poder utilizar el puerto serie, debemos ejecutar una modificación en el proceso de boot para desactivar la consola serie, que de otra manera esta utilizaría el puerto serie.

DESACTIVAMOS LA CONSOLA SERIE

Casi todos los sistemas GNU/Linux activan al inicio la funcionalidad consola, es decir, la posibilidad de conectarse a la línea serie con un terminal - real o virtual - para gestionar todo el sistema a través de esta interfaz de comunicación. Un recuerdo de los tiempos en los que se conectaba a los ordenadores vía modem utilizando las líneas telefónicas normales. A veces es aún la única posibilidad de acceso a los siste-

Fig. 3



Listado 1

```
/*
*  GSMAccendi
*/

//Include ArduPi library
#include "arduPi.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOn(){
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
}

void resetModule(){
    digitalWrite(resetModulePin,HIGH);
    delay(500);
    digitalWrite(resetModulePin,LOW);
    delay(100);
}

int main (){
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf ("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
       {
       printf ("uno\n");
       resetModule();
       delay(2000);
       }
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
       {
       printf ("due\n");
       switchModuleOn();
       }

        return (0);
}
```

mas embebidos o servidores. Para deshabilitar la consola utilizamos WINScp, vamos a la carpeta "boot" con el comando:

```
cd /boot
```

y abrimos, con doble clic el archivo /boot/cmdline.txt y eliminamos, prestando atención, los parámetros de configuración (**Fig. 4**):

```
console=ttyAMA0,115200 kgdboc=ttyAMA0,115200
```

Salvamos el archivo y lo cerramos.

EJEMPLOS DE PROGRAMAS (ON/OFF DEL MÓDULO Y ENVÍO SMS)

Veamos la lógica del programa de encendido del módulo mirando el **Listado 1**. Como primera consideración podéis verificar que los programas están codificados para ejecutar una única secuencia de instrucciones, contraria a los programas para Arduino, que funcionan infinitamente en el interior de un ciclo loop(). Esto permite utilizar los programas individualmente como módulos de una aplicación de nivel superior.
El proceso de encendido del módulo GSM prevé los siguientes pasos:

- Control, mediante el envío en serie de un comando "AT" para verificar que el módulo no esté ya encendido;
- En caso de respuesta negativa (al envío de un comando AT, en situaciones normales se recibe como retorno el paquete de datos "OK"), como control adicional, se ejecuta el proceso de reset con el sucesivo envío de un segundo comando "AT";
- En caso de otra respuesta negativa se ejecuta el proceso de encendido que consiste en llevar a nivel alto el pin GPIO27 durante dos segundos.

El objetivo de los controles preliminares es debido al hecho que el proceso de encendido y apagado es el mismo, entonces, si se ejecuta el programa de encendido sin controles y el módulo GSM estaba ya encendido, se apaga. Para compilar el programa GSMEnciende se utiliza el comando:

```
g++ -lrt -lpthread GSMEnciende.cpp
arduPi.o -o GSMEnciende
```

Para ejecutar el programa, después de habernos posicionado en la carpeta ArduPi con el comando:

```
cd /home/ArduPi
```

teclea:

```
./GSMEnciende
```

Podemos seguir el proceso de encendido del módulo GSM observando el LED verde, al arrancar permanece encendido alrededor de dos segundos, después de apaga y empieza a parpadear con intervalos de alrededor de medio segundo, señal de que el módulo está buscando la red GSM. Cuando la red está enganchada el LED parpadea con una frecuencia mucho más baja, alrededor de un parpadeo por segundo.
En el **Listado 2** vemos el programa para el apagado del módulo, que expresa una lógica similar al programa anterior:

- Control, mediante el envío en serie de un comando "AT" para verificar que el módulo no esté ya apagado;
- En caso de respuesta negativa, como control adicional, se ejecuta el proceso de reset con el sucesivo envío de un segundo comando "AT";
- En caso de respuesta positiva se ejecuta el proceso de apagado que consiste en llevar el pin GPIO27 a nivel alto durante dos segundos.

También en este caso el objetivo de los controles preliminares se

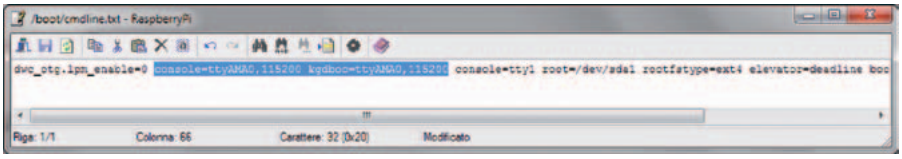


Fig. 4

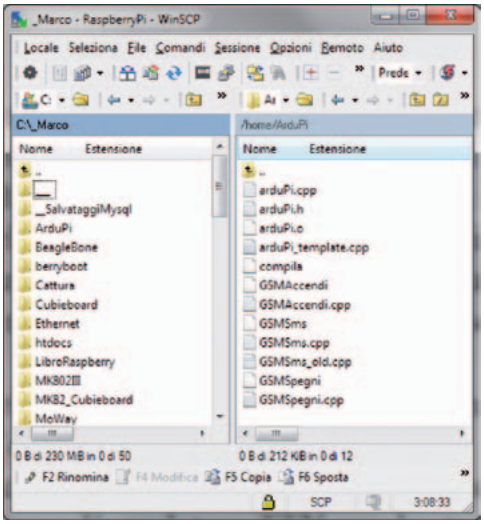


Fig. 5

debe al hecho de que el proceso de encendido y apagado es el mismo, entonces, si se ejecuta el programa de apagado sin controles y el módulo GSM está ya apagado, se enciende. Para compilar el programa GSMSpengi se utiliza el comando:

```
g++ -lrt -lpthread GSMApaga.cpp
arduPi.o -o GSMApaga
```

Para ejecutar el programa, siempre después de habernos posicionado en la carpeta ArduPi, se utiliza el comando:

```
./GSMApaga
```

En el **Listado 3** vemos el programa que permite enviar un SMS.

- En la variable phone_number[] se inserta el número de teléfono a llamar;
- AT+CMGF=1 establece la modalidad SMS en formato texto;
- AT+CMGS= envía el mensaje SMS establecido en la instrucción siguiente;
- Finalmente se envían los caracteres CTRL-Z para cerrar la llamada.

Para compilar el programa se usa el comando:

```
g++ -lrt -lpthread GSMSms.cpp
arduPi.o -o GSMSms
```

Para ejecutar el programa se ejecuta el comando:

```
./GSMSms
```

Antes de ejecutar el programa es necesario ejecutar el programa de arranque del módulo GSM, para asegurarse de que el módulo esté efectivamente encendido y funcionando.
Cuando se vaya a desarrollar la aplicación de control, será tarea de esta última gestionar la secuencia correcta de llamada de los módulos, con los relativos controles de estado y gestión de las condiciones anómalas.
En la **Fig. 5** es visible el estado de la carpeta después de la escritura y compilación de los programas.
En la **Fig. 6** es visible la secuencia de los comandos de compilación y de ejecución del arranque del módulo GSM, envío SMS y apagado del módulo.
Veamos ahora la segunda modalidad, aquella que permite de interactuar directamente con el puerto serie mediante el programa de emulación TTY "CuteCom".
Y puesto que hemos asignado la Raspberry Pi para ser gestionada en remoto a través protocolo SSH, continuamos en la misma dirección de manera que utilizamos el emulador TTY con las mismas modalidades.

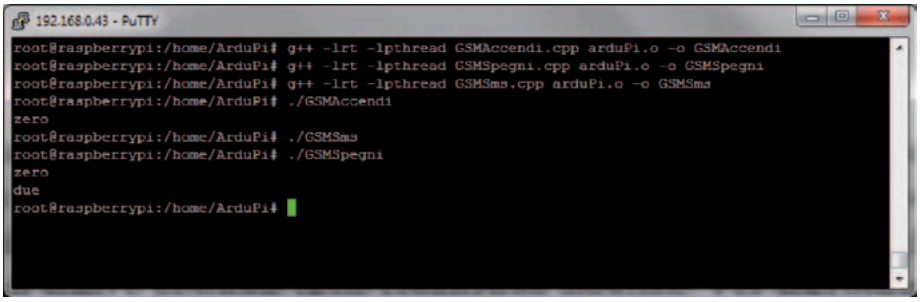


Fig. 6

INSTALAMOS EL EMULADOR TTY SERIE

Para la gestión del puerto serie hemos elegido utilizar el programa "CuteCom". Lo instalamos con los comandos comunes:

Listado 2

```
/*
*  GSMSpegni
*/

//Include ArduPi library
#include "arduPi.h"

int resetModulePin = 9;
int onModulePin = 8;

void switchModuleOff(){
    digitalWrite(onModulePin,HIGH);
    delay(2000);
    digitalWrite(onModulePin,LOW);
}

void resetModule(){
    digitalWrite(resetModulePin,HIGH);
    delay(500);
    digitalWrite(resetModulePin,LOW);
    delay(100);
}

int main (){
    Serial.begin(115200);
    delay(2000);
    pinMode(resetModulePin, OUTPUT);
    pinMode(onModulePin, OUTPUT);
    Serial.flush();
    printf ("zero\n");
    Serial.print("AT");
    delay(1000);
    if (Serial.available()==0)
       {
       printf ("uno\n");
       resetModule();
       delay(2000);
       }
    Serial.print("AT");
    delay(1000);
    if (Serial.available()>0)
       {
       printf ("due\n");
     switchModuleOff();              //
       }

        return (0);
}
```

Listato 3

```
/*
*  GSMSms
*/

//Include ArduPi library
#include "arduPi.h"

int timesToSend = 1;        // Número de SMS a enviar
int count = 0;
int ok = 0;
int numCar = 0;

char phone_number[]="3........9";  // * Número teléfono dest. SMS

void setup(){
    Serial.begin(115200);   // UART baud rate
    delay(2000);
    Serial.println("AT+CMGF=1");              // pone el modo SMS a texto
    delay(100);
}

void loop(){
    while (count < timesToSend){
        delay(1500);
        Serial.print("AT+CMGS=\"");   // envía el núero de SMS
        Serial.print(phone_number);
        Serial.println("\"");
        while(Serial.read()!='>');
        Serial.print("Se arriva funziona"); // envía el cuerpo del SMS
        delay(500);
        Serial.write(0x1A);          // envía ++
        Serial.write(0x0D);
        Serial.write(0x0A);
        delay(5000);
        count++;
    }
}

int main (){
         setup();
         while(1){
                  loop();
               if (count == timesToSend)
               {
               break;
               }
         }
         return (0);
}
```

```
apt-get update
apt-get upgrade
apt-get install cutecom
```

INSTALAMOS XMING

Xming es un servidor X open source para Windows que permite visualizar en el escritorio del PC Windows las ventanas gráficas de las aplicaciones que se ejecutan en el sistema GNU/Linux en remoto.

No se transfiere el escritorio entero, solo únicamente la o las ventanas gráficas de aplicaciones individuales que puede ser posicionado a voluntad en el escritorio del PC local, junto a las ventanas del PC mismo o de otros sistemas remotos.

Podéis imaginar las anécdotas surgidas en este sentido en el mundo de los administradores de sistema, sobre los comandos dados mirando la ventana de administración de un sistema y tecleando en aquella de un sistema distinto. Aun así la funcionalidad es fascinante.

X es la aplicación basada sobre el modelo cliente/servidor que permite exportar sobre una máquina el front-end gráfico de una aplicación en ejecución en un sistema distinto.

Se apoya sobre el protocolo SSH y sobre la aplicación Putty (o Kitty).

La denominación cliente/servidor suele confundir a los usuarios de X porque los términos parecen invertidos: sobre el escritorio Windows del usuario está en ejecución el "servidor" que es utilizado por programas de aplicaciones (cliente). X proporciona un servicio de visualización a los programas, desde este punto de vista actúa como servidor, mientras el programa de aplicación (que puede también ser gestionado a través de conexión remota) utiliza los servicios, actuando como un cliente.

El protocolo de comunicación entre servidor y cliente trabaja en modo transparente respecto a la red: ambos pueden estar sobre la misma máquina o sobre máquinas distintas, también con arquitecturas y sistemas operativos diferentes; el servidor y el cliente pueden también comunicar en modo seguro a través de la red explotando un túnel cifrado (normalmente se suele utilizar SSH).

El servidor X puede ser:

- Un programa de sistema que controla la salida video de un ordenador;
- Un componente hardware dedicado (los llamados terminal X: ordenadores dotados solo del hardware necesario para ejecutar el servidor X, pensa-

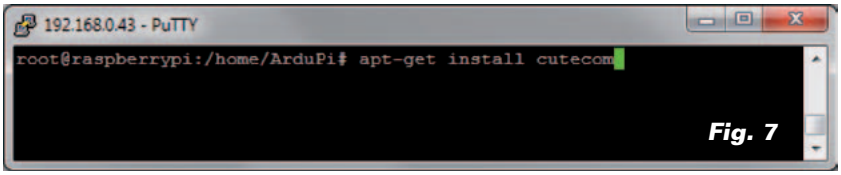


Fig. 7

Fig. 8

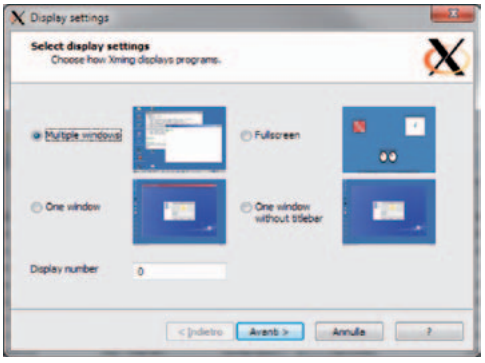


Fig. 9

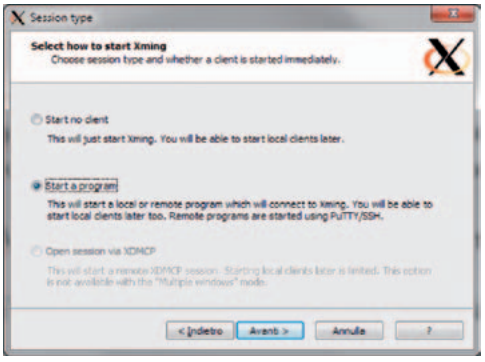
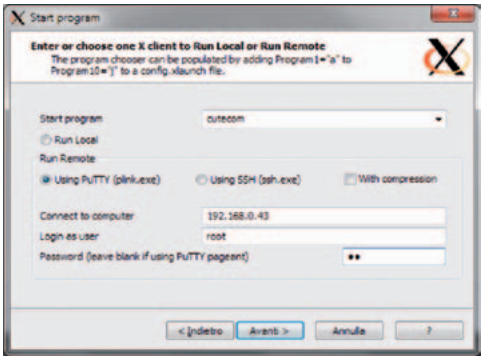


Fig. 10



dos para visualizar aplicaciones ejecutadas en servidores de aplicaciones específicas);

- Una aplicación que muestra datos sobre una ventana de otro sistema gráfico.

Xming es descargable en:
http://sourceforge.net/projects/xming/
Lo descargamos e instalamos como cualquier aplicación de Windows.
Nuestra atención se dirige al programa XLaunch, lo encontramos en el menú de Windows en la elección Xming, que nos permite de lanzar en remoto cuantas aplicaciones individuales queramos en ventanas separadas de Windows. Una vez instalado Xming lanzamos CuteCom en remoto sobre el escritorio de nuestro PC. Abrimos XLaunch (**Fig. 8**). Seleccionamos "Multiple Windows" y "Adelante". En la siguiente pantalla seleccionamos "Start a program" (**Fig. 9**) y otra vez adelante. Ahora seleccionamos "Using Putty" (**Fig. 10**). En el campo Start Program tecleamos "cutecom", en "Connect to computer" insertamos la dirección IP de la Raspberry Pi. En "Login as user" tecleamos "root" y en el campo password la contraseña correspondiente. De nuevo "Adelante" y "Adelante" también en la página siguiente (**Fig. 11**). Llegamos a la última página donde confirmamos con "Fin" (**Fig. 12**). Damos "OK" y "NO" en los sucesivos mensajes y ... esperamos. Al final obtenemos la ventana de la **Fig. 13**.

GESTIONAMOS EL MÓDULO GSM DESDE EL EMULADOR TTY

La ventana de CuteCom está dividida en "zonas" funcionales. En alto están presentes los campos y los pulsadores para configurar, activar y desactivar la comunicación serie, abajo el campo de texto para insertar los comandos a enviar sobre el puerto serie, un área con la "history" de los comandos enviados y algunos comandos de configuración que permiten:

- Activare y desactivar un log;
- Añadir al comando caracteres de control como el final de línea;
- Transmitir y/o recibir los datos en formato ascii o hexadecimal.

Finalmente en el centro podemos ver el área donde se evidencian los mensajes de llegada del puer-

Fig. 11

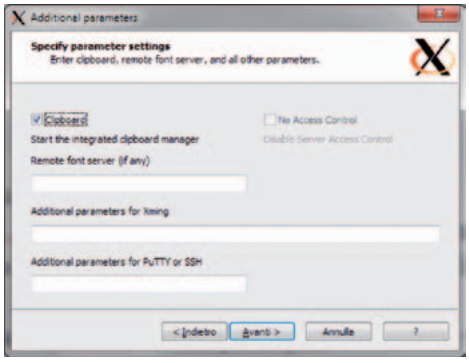


Fig. 12

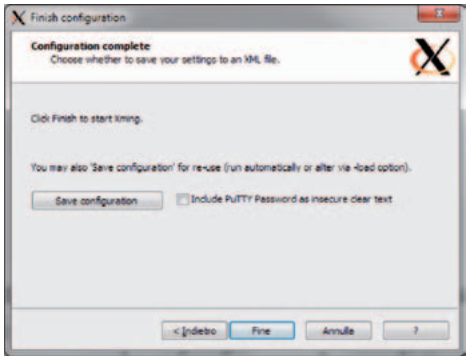
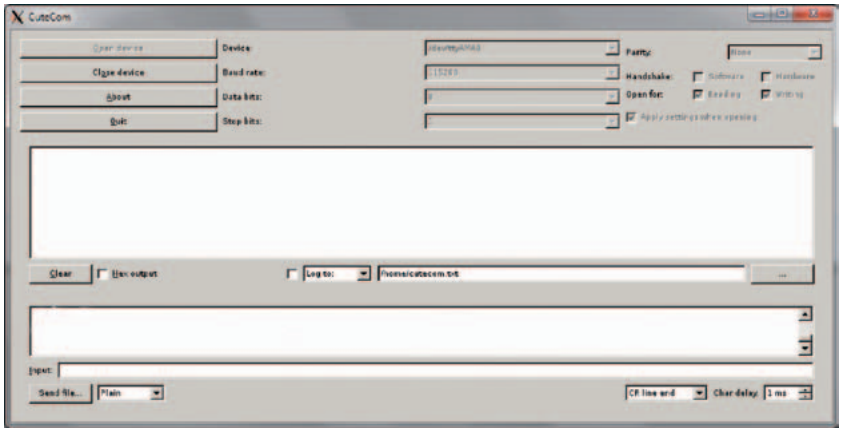


Fig. 13



to serie. La primera operación a realizar es configurar el nombre del dispositivo (y su ruta relativa) del puerto serie al cual queremos conectarnos, en nuestro caso:

```
/dev/tty/AMA0
```

Si no la encontramos en el menú desplegable de los valores predeterminados, la escribimos directamente en el campo de configuración. Configuramos el puerto asignando 115200 para la velocidad, 8 para la longitud de los caracteres, 1 para el "Stop bit" y "none" para la paridad. No seleccionamos alguna opción para el protocolo (*handshake*). Marcamos sin embargo ambos check box para la apertura en lectura y escritura. Dejamos sin marcar los check box "Hex output" y "Log to". En el campo de texto cercano a "Send file" seleccionamos "Plain", en el penúltimo campo abajo a la derecha seleccionamos "CR line end" y en el último "1 ms". Bien, podemos empezar. Pero primero asegurémonos que el módulo SIM9XX tenga una SIMCard insertada, esté encendido y conectado a la red móvil comprobando que el LED parpadea lentamente. En caso contrario lo encendemos ejecutando el programa:

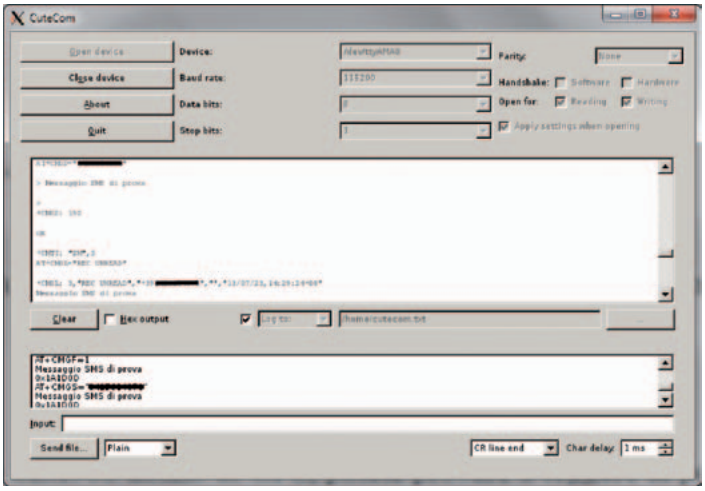
```
./GSMEnciende
```

Ahora podemos pulsar "Open Device" arriba a la izquierda en la ventana de CuteCom. Si todo está en orden veremos volvere gris el área de configuración arriba y podremos empezar a enviar comandos insertándolos en el campo apropiado abajo. Empezamos con "AT", lo tecleamos en el campo apropiado y pulsamos "Send file", deberemos recibir en respuesta "OK", en caso contrario podría ser que el módulo non se encuentre en modo comandos. Activamos el "Command Mode" con el comando "+++".
Haciendo referencia al manual para el módulo que estamos utilizando SIM900 o SIM908, podemos probar el efecto de cada comando, sea para la configuración del módulo que para la ejecución de secuencias de comandos operativos.
Algún ejemplo. Conectamos un altavoz autoalimentado a la salida audio y un micrófono a la entrada MIC o utilizamos un auricular con micrófono con los pin separados. Probamos a ejecutar y a recibir una llamada.
Primer damos al comando:

```
AT+CALM=0
```

que permite activar el sonido (es posible también aplicar la musiquita, consultar el manual). Ahora

Fig. 14



damos al comando :

```
ATD<numero_de_telefono> seguido de ;", por ejemplo
ATD+393.......7;.
```

Cuando el teléfono llamado acepta la llamada podremos hablar con nuestro interlocutor, para interrumpir la llamada de nuestro lado damos al comando:

```
ATH
```

Si un teléfono externo nos llama, sentiremos el sonido en el altavoz y a cada tono veremos la escritura "RING" en el área central. Para responder a la llamada teclearemo el comando:

```
ATA
```

Para enviar un SMS tecleamos la secuencia de comandos:

```
AT+CMGF=1
AT+CMGS="<numero_de_telefono>"
```

recordando insertar el número de teléfono entre corchetes. Atendemos al mensaje de retorno

```
">"
```

Tecleamos el texto del mensaje, por ejemplo:

```
Mensaje SMS de prueba
```

esta vez sin corchetes, para terminar seleccionamos en el penúltimo campo abajo a la derecha, "Hex input" e insertamos:

```
Ox1A1D0D y envío
```

(equivale a CTRL-Z para cerrar el mensaje SMS).

Para proseguir recordamos reportar el valor del penúltimo campo abajo a la derecha al valor "CR line end". Para leer la lista de los mensajes SMS recibidos y aún no leídos:

```
AT+CMGL="REC UNREAD"
```

Obtenemos (**Fig. 14**):

```
+CMGL: 3,"REC UNREAD","+343........1","","13/07/23,
14:29:24+08"
Mensaje SMS de prueba
```

donde "3" es el índice del mensaje. Podemos leer el mensaje solo, utilizando el índice, también con el comando:

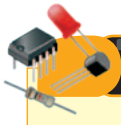
```
AT+CMGR=3
```

Finalmente, si habéis instalado un módulo SIM908 dotado de GPS, podéis probar a leer la posición (con la antena al aire libre) utilizando el comando:

```
AT+CGPSINF=0
```

Obtendréis como resultado un paquete de datos compuesto por los campos modo, longitud, latitud, altura, fecha/hora, TTFF (Time To First Fix), número de satélites, velocidad y dirección de brújula. Podéis elegir utilizar los comandos nativos para diseñar las funcionalidades que podréis después codificar en programas estructurados. Profundizaremos la realización de aplicaciones estructuradas y modulares en los próximos artículos.

(179025)



el MATERIAL

El shield de expansión GSM/GPS/GPRS para Raspberry Pi está disponible en kit de montaje al precio de 13,90 Euros (cod. FT1075K). El kit no incluye el módulo GSM que puede adquirirse por separado (cod. FT900M, 49,00 Euros) ni la antena (cod. ANTGSMSTL-F01, 8,00 Euros). Y también es posible utilizar el shield con el módulo GSM/GPS Cod. FT971 (65,00 Euros) al cual se conectan las dos antenas (ANTGPS-SMA, 14,00 Euros y ANTGSMSTL-S01, 8,00 Euros) mediante dos conectores Cod. CAVOUFLSMA, 6,00 Euros cada uno.

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