ARDUINO YÚN:
VIDEOVIGILANTE ELECTRÓNICO

Conectamos una cámara de video y aprovechamos las nuevas funcionalidades de Arduino Yún para realizar un "perro guardián" electrónico, que nos permita tener bajo control nuestras cosas, un ángulo oculto del jardín, o nuestro vehículo aparcado en el patio.

MARCO MAGAGNIN

En la edición anterior de la revista os presentamos la "penúltima" tarjeta de la familia Arduino. La combinación de las dos "almas" presentes en la tarjeta Arduino Yún, constituidas por la tradicional de Arduino y la de GNU/Linux, permite ampliar notablemente la gama de aplicaciones que se pueden afrontar con la sencillez típica de la filosofía Arduino. En esta ocasión hemos querido abordar un proyecto que jamás habría podido realizarse antes con una tarjeta Arduino. Conectar una webcam, poder visualizar sobre un navegador web las imágenes trasmitidas directamente en "streaming" por un servidor web presente sobre la tarjeta misma y activar acciones si "algo" o "alguien" se mueve en el área del campo de visión que queremos tener bajo control. Imaginemos el caso de un patio, quizás con nuestro automóvil aparcado en el interior. En general en este área no debería moverse nadie, en particular alrededor del lugar

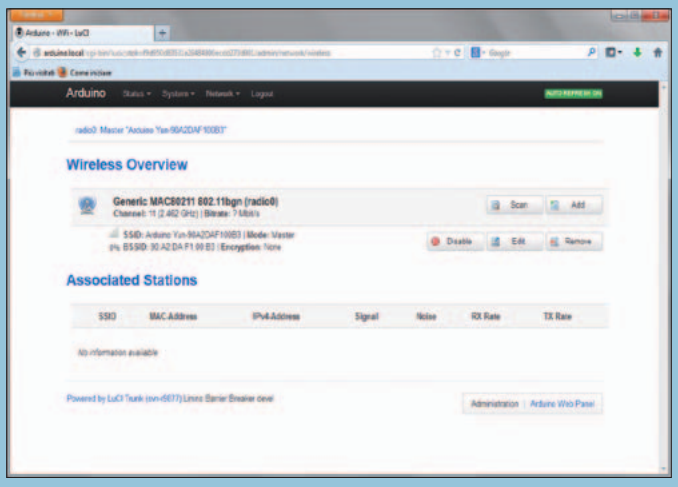


Fig. A

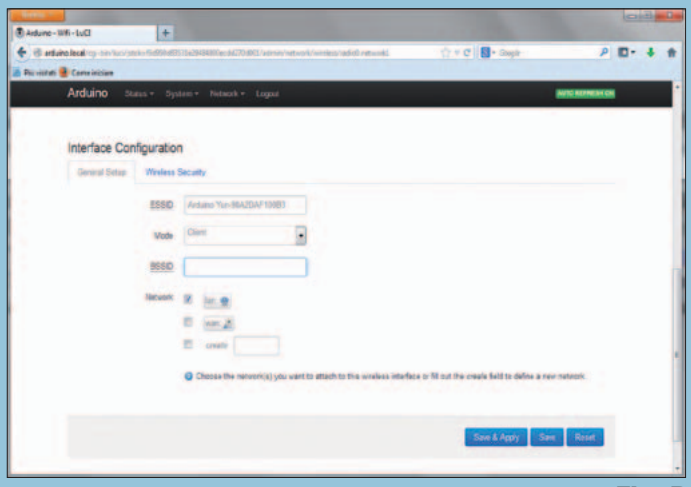


Fig. B

Para la administración del sistema operativo Linino está disponible el instrumento LuCI, heredado del progenitor OpenWRT. Entre las distintas funcionalidades está disponible el panel de configuración de interfaz WiFi, inicialmente configurada como Access Point. Se entra en el instrumento de administración simplemente mediante un buscador web escribiendo la dirección "arduino.local". Se pide el password "arduino". En la página de estado de las conexiones hacemos clic sobre el pulsador "Configure" y en la página siguietne clic sobre el link "advanced configuration panel (LuCi)". Nos encontramos en el instrumento de administración LuCi. En esta pági-

donde está aparcado el vehículo. Con este proyecto podremos, por ejemplo, encuadrar el vehículo con la cámara y seleccionar el área que encuadra la puerta como área "sensible". Si cualquier cosa o alguien se mueven en esta zona lanzará un proceso de alarmas para avisarnos del evento. Después de comprobarse el evento podremos decidir después enviar un tweet de aviso, justo para reutilizar un ejemplo ya propuesto en la edición anterior de la revista, la 320, o un e-mail, como describimos en este artículo, o también acciones combinadas, como las propuestas y además del encendido de un LED de notificación. También un pequeño proyecto como este vuelve a proponer la diferencia de enfoque a tener en cuenta al proyectar aplicaciones para una tarjeta embebida frente a aquellas para una tarjeta Arduino clásica. Una tarjeta Arduino clásica requiere el típico enfoque de microcontrolador, o la proyección de una aplicación "monolítica" constituida por un único programa, "sketch" en el caso de Arduino, que una vez cargado en la memoria de la tarjeta personaliza el comportamiento. La atención debe estar puesta en definir la secuencia correcta de las operaciones, respetar las modalidades de interfaz de los sensores y en la definición de los tiempos de manera que se obtengan mediciones correctas. La conectividad con el exterior es insuficiente (aunque la disponibilidad de shields Ethernet, WiFi y Bluetooth han mitigado este aspecto) y es casi imposible ejecutar simultáneamente funciones de control de sensores y servicios web tan exigentes como la gestión y la transmisión de imágenes. Distinta es la situación para un sistema embebido con dos procesadores dedicados como en Arduino Yún. En este caso la atención está puesta en el diseño completo de la arquitectura que se quiere realizar, en particular a la subdivisión en módulos de la aplicación, en la colocación de cada módulo sobre el procesador más apropiado y a la interacción y comunicación entre ambos. La aplicación se apoya sobre una infraestructura ya existente, constituida por el sistema operativo, en nuestro caso Linino, y por las colecciones de programas listos para desarrollar muchas de las funciones que realizaremos. Una vez diseñada la aplicación, la primera operación a realizar es la elección de componentes ya disponibles, que nos pueden ser útiles. Escribiremos código nuevo solo para las partes estrictamente específicas de la aplicación que no podemos encontrar ya listas. Este enfoque es posible gracias a la presencia del sistema operativo GNU/Linux, y de las colecciones de programas disponibles bajo varias licencias de distribución open source. Esto es verdad también para la sección Arduino de la tarjeta, por la cual están disponibles tantísimas librerías y ejemplos de aplicaciones ya listas para usar. Debemos poner atención también a la comunicación entre los distintos componentes de la aplicación, que son competencia de los procesadores: siempre un solo programa para Arduino mientras puede haber

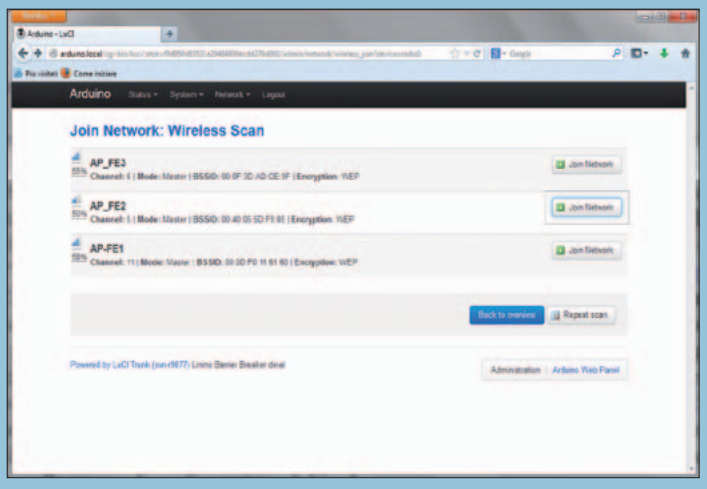


Fig. C

Fig. D

na seleccionamos el menú "Network" y el submenú "Wifi". En la página que se puede ver en la Fig. A, clic sobre el pulsador "Edit" y, en la página de la Fig. B, bajamos hasta la sección "Interface configuration" y modificamos el campo mode, de "Access Point" a "Client", utilizando la selección desplegable. Salvamos haciendo clic sobre "Save & Apply". Seleccionamos de nuevo el menú "Network" y el submenú "Wifi" y esta vez pulsamos sobre "Scan". Nos aparece el listado de las redes WiFi disponibles (Fig. C). Seleccionamos aquella a la que queremos conectarnos e insertamos las credenciales de acceso en la página de configuración de Fig. D.

varios a la vez en la sección GNU/Linux. Debemos entonces diseñar la interfaz de comunicación entre los procesos, actividad que debe ser coordinada tanto en base a las exigencias aplicativas, como teniendo en cuenta la necesidad de evitar situaciones de conflicto, por ejemplo en el uso de recursos compartidos.

DESCRIPCIÓN DEL PROYECTO

Pero procedamos en orden, lo que queremos realizar es visible en el esquema de la **Fig.1**: un sistema de monitorización de un área a "proteger". En caso de detección de un movimiento en el área bajo vigilancia, queremos que el sistema nos avise en remoto con el envió de un e-mail y en local con el encendido de un LED. Las imágenes que "inmortalizan" el intruso deben ser memorizadas en una carpeta sobre la SDCard de la tarjeta para una visualización a "posteriori". Un pulsador permite "restablecer" la notificación visual vía LED. La webcam está conectada al puerto USB del procesador Atheros AR9339.

Como software para la gestión en "streaming" de las imágenes tomadas por la cámara, para la detección del movimiento y archivo de los fotogramas correspondientes utilizamos el paquete "motion", realizado específicamente para el reconocimiento del movimiento. Para que el sistema operativo Linino reconozca la cámara, debemos cargar también el driver para el periférico de video. Nos hemos ocupado ya de la gestión de los periféricos y de los relativos drivers en GNU/Linux, en ediciones anteriores de la revista. La

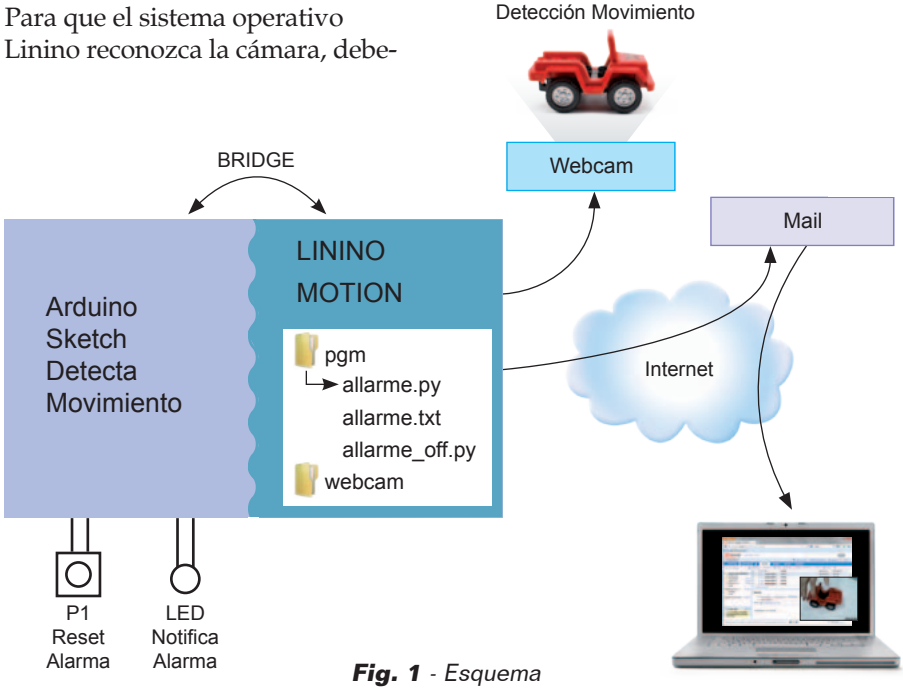
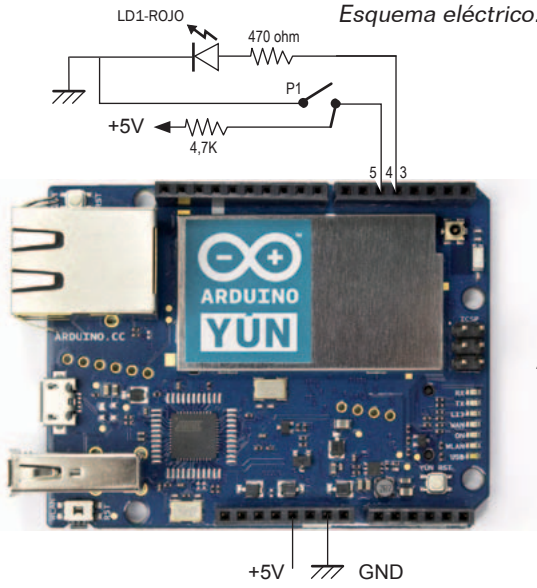


***Fig. 1** - Esquema de la aplicación.*

Fig. 2
Esquema eléctrico.



particularidad en Linino está en la posibilidad de añadir drivers desde el exterior, sin necesidad de tener que reconfigurar y recompilar los módulos del kernel de GNU/Linux. El paquete "motion" nos permite realizar el streaming de las imágenes de la cámara, que hacemos visible vía navegador de Internet. Cada vez que se detecta un movimiento en el área bajo control, queremos que se registren una serie de fotogramas del intruso y queremos además que el sistema nos avise del evento tanto con el envió de un e-mail a nuestra dirección, como con el encendido de un LED, que permanece en ese estado hasta que no lo apaguemos accionando un pulsador. También es posible enviar un tweet en el caso de alarma aprovechando el sketch (con las debidas modificaciones) que presentamos en la edición 320 de la revista. Cuando se detecta un movimiento, la función "motion" activa la ejecución de un programa en entorno Linino que procede a ejecutar las acciones de notificación. El programa "allarme.py" como primera acción de notificación modifica el contenido de un archivo "allarme.txt" llevándolo del valor "0" a "1". Una especie de "semáforo". Después procede al envió del e-mail a la dirección especificada adjuntando el ultimo fotograma generado por la acción detectada. Para la detección de la notificación por parte del sketch Arduino utilizamos las funcionalidades disponibles por la arquitectura Bridge. El sketch en ejecución sobre Arduino, utilizando la librería "FileIO" lee periódicamente el contenido del archivo "allarme.txt" y, si encuentra el valor "1", enciende el LED rojo conectado al pin 4 de Arduino. Siempre si el valor en el archivo es "1" el sketch, utilizando esta vez la funcionalidad "Process", requiere la ejecución de un programa sobre Linino, "allarme_off.py" que repone a cero el contenido del archivo. El LED rojo queda encendido hasta que no se apaga voluntariamente accionando el pulsador conectado al pin 5 de Arduino.

DESARROLLO PRÁCTICO

El desarrollo de este proyecto nos permite profundizar algunos aspectos de la administración del sistema operativo Linino. Antes sin embargo vamos a realizar el circuito simple visible en la **Fig. 2**. Nosotros hemos utilizado el circuito sobre una breadboard usado en el ejemplo de la edición 320 de la revista, del que simplemente hemos eliminado el LED verde. Siempre para realizar este proyecto es necesario que la tarjeta Arduino Yún este en la configuración prevista en el citado ejemplo de la edición 320, con una tarjeta microSD insertada en la cual estén presentes las carpetas

/arduino y
/data

Volvamos a la configuración "software". El instrumento a nuestra disposición es LuCi, la aplicación con interfaz web, que proporciona un menú de fun-

Fig. 3

Fig. 4

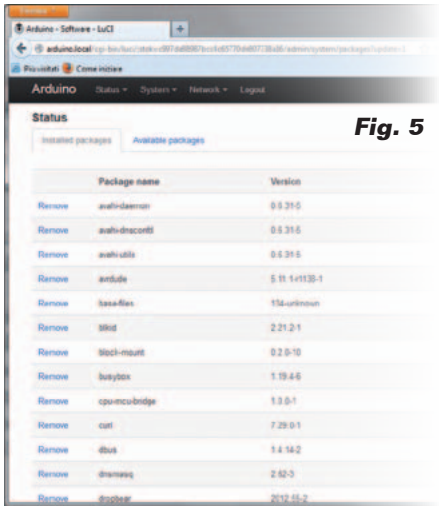


Fig. 5

cionalidad de administración, subdividido por argumentos, de los distintos aspectos del sistema operativo Linino, entre los cuales está el instrumento de gestión de los paquetes. En los artículos dedicados a la tarjeta Raspberry Pi hemos gestionado siempre la instalación de paquetes utilizando la interfaz de línea de comandos. Obviamente también en Linino podemos utilizar la interfaz en línea de comandos, utilizando el gestor de paquetes opkg, (open package manager), el equivalente de APT de Raspberry Pi, pero está bien introducir también los instrumentos dotados de interfaz gráfica que, cuando están disponibles y bien hechos, simplifican la vida a los usuarios.

Una aclaración antes de comenzar; en Arduino Yún el sistema operativo Linino, como su progenitor OpenWRT está instalado sobre la memoria NAND del dispositivo. Una memoria de 64Mb de capacidad que deja poco espacio a la carga de paquetes adjuntos. Para poder extender la memoria donde cargar los paquetes es necesario una intervención de hacking de la tarjeta que os describiremos en un próximo artículo. Por ahora contentémonos con cargar los nuevos paquetes en la memoria NAND. Conectamos la tarjeta a nuestra red a través de cable Ethernet o por conexión WiFi, pero en este caso debemos configurar la tarjeta como cliente WiFi, como se describe en el recuadro "Configuración WiFi" de Arduino Yún. En cada caso asegurémonos que Arduino Yún esté conectado a la misma red de nuestro PC. Esperamos un rato, el boot completo de Arduino Yún dura alrededor de medio minuto, más largo incluso que el de una tarjeta normal Arduino. Abrimos un navegador de Internet y escribimos la dirección: *http://arduino.*

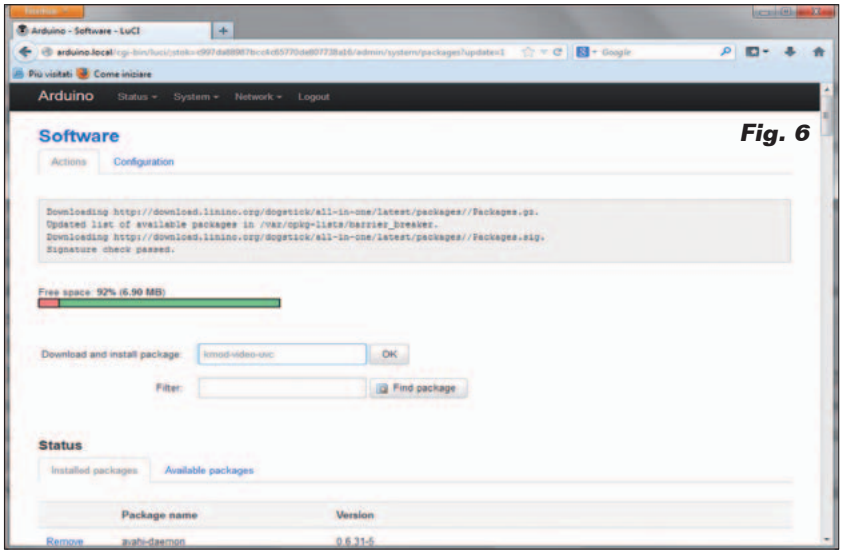


Fig. 6

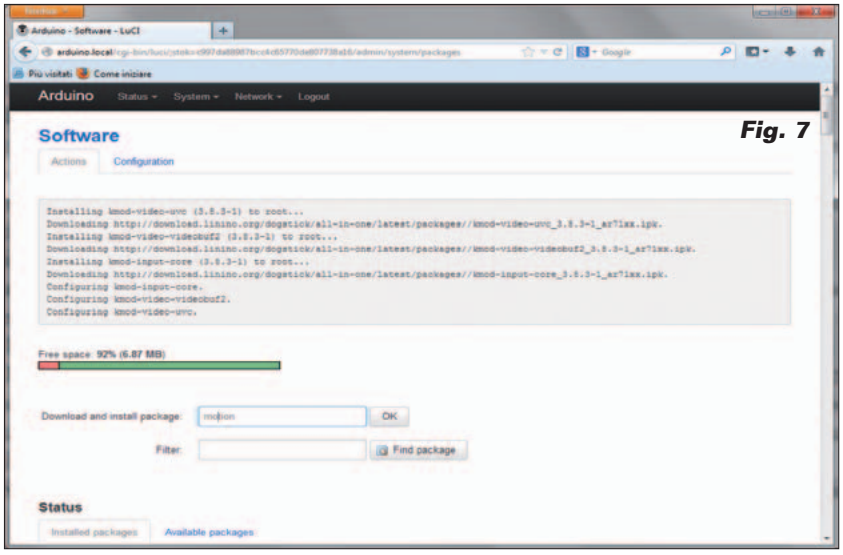


Fig. 7

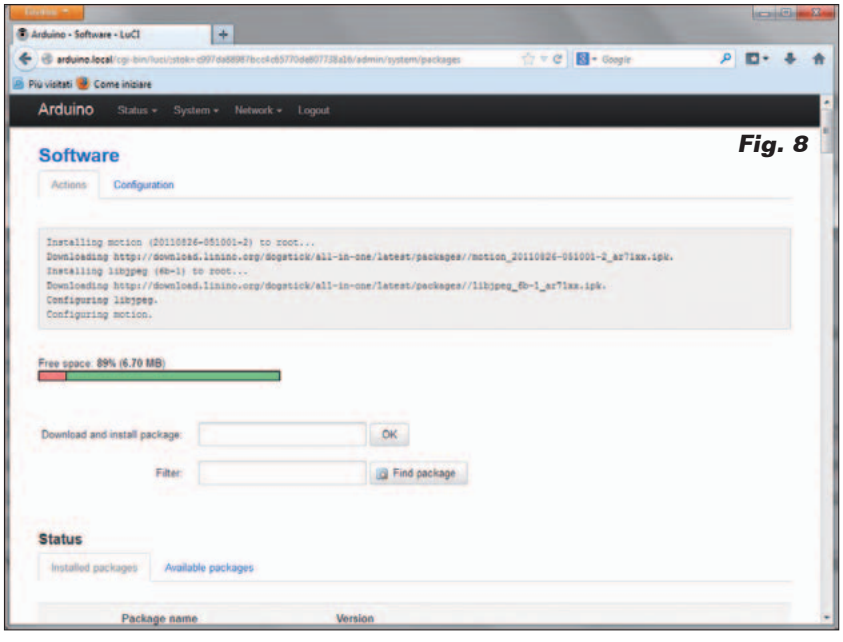


Fig. 8

Fig. 9

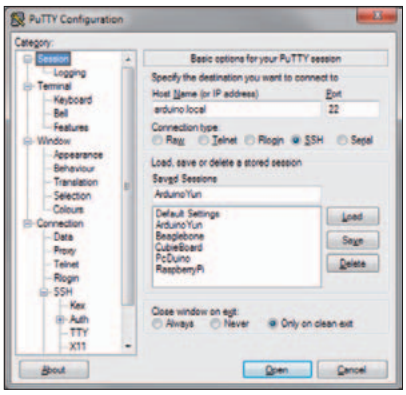


Fig. 10

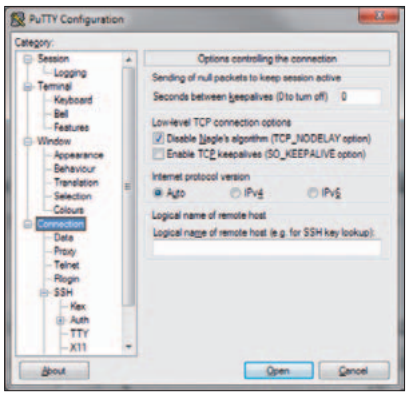
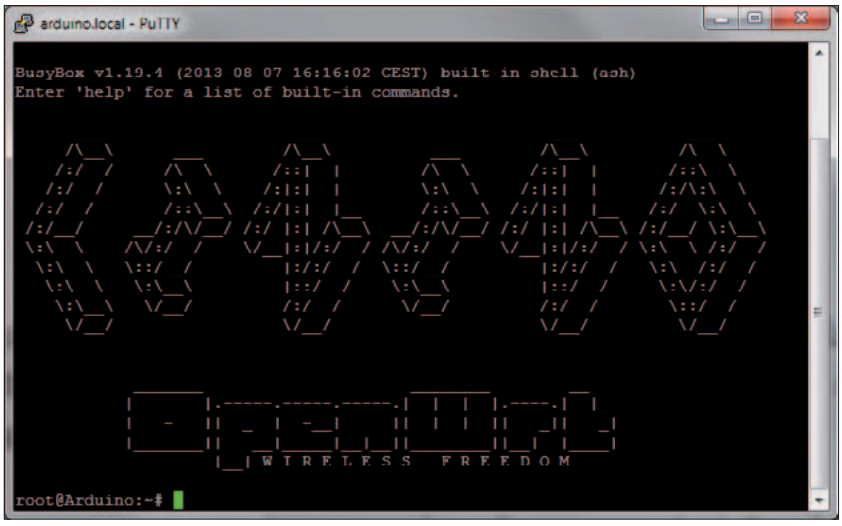


Fig. 11



local.

Escribimos el password "arduino" y entramos en la aplicación LuCI. Hacemos clic sobre el pulsador azul "Configure" y después, en la página siguiente, sobre el link "advanced configuration panel (luci)". Nos aparece la pantalla de la **Fig. 3,** donde desde el menú "System" seleccionamos la opción "Software" **(Fig. 4)**. En este modo entramos en la página de gestión de los paquetes software. Podemos ver el espacio libre para la instalación de nuevos paquetes. Por curiosidad movemos la página hacia abajo, donde podemos ver la lista de los paquetes ya instalados (**Fig. 5**).

Se habéis seguido nuestros artículos dedicados a Raspberry Pi reconoceréis muchos de los paquetes instalados. Primero actualizamos el listado de paquetes disponibles clicando sobre el pulsador "Update lists", el equivalente del comando "opkg update" dado por la línea de comando. Obtenida la actualización del catálogo de paquetes disponibles, instalamos el driver para la webcam. Insertamos el nombre del paquete "kmod-video-uvc"en el campo de texto a derecha de la etiqueta "Download and install package" y clic sobre el pulsador "OK" (**Fig. 6**). Al terminar obtenemos la página de la **Fig. 7**. Con el mismo proceso instalamos el paquete "motion". Terminada también esta instalación (**Fig. 8**) debemos configurar el paquete. Con tal fin, debemos entrar en el corazón del archivo system, y el mejor modo es utilizar el servicio SSH (Secure Shell) que es el servicio de acceso remoto normalmente utilizado por los usuarios GNU/Linux para administrar los sistemas conectándose desde un PC externo. El servicio está activo de forma predefinido sobre Linino, mientras que para utilizarlo desde PC debemos instalar los correspondientes programas cliente. Recordemos solo que los paquetes a encontrar son Putty y WinSCP. Se encuentran con una simple búsqueda en la web, son open source y de libre uso. Una vez instalados son configurados para conectarse a Arduino Yún. Lo primero es lanzar el programa Putty, lo configuramos como visible en las **Fig. 9** y **Fig. 10** y nos conectamos a Arduino Yún. Nos piden las credenciales de login. Escribimos "root" en la petición de login y respondemos con "arduino" a la petición de la password. Obtendremos la página de bienvenida de la shell de Linino visible en **Fig. 11**.

Fig. 12

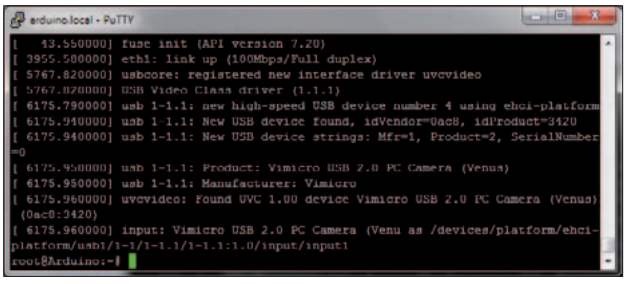
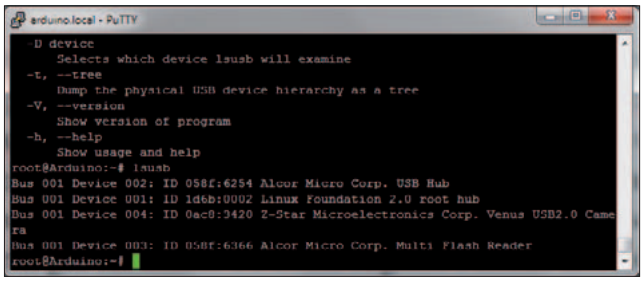


Fig. 13



Ahora conectamos webcam a la toma USB-A (la grande), que encabeza el procesador Atheros que alberga Linino. Damos al comando:

dmesg

Al final de la lista vemos si la webcam ha sido reconocida, como se ve en la **Fig. 12**. Ahora damos al comando:

lsusb

y comprobamos que la webcam es reconocida como device USB (**Fig. 13**).
Podemos controlar con que nombre de archivo (archivo de device) es reconocida la webcam, en general "video0", lanzando el instrumento WinSCP, el archivo manager que utiliza el protocolo SSH. Configuramos WinSCP como se puede ver en las **Fig. 14, 15, 16 y 17.** Vamos a la carpeta /dev y buscamos un archivo con nombre *video0:* se trata del archivo de device que identifica la webcam (**Fig. 18**).
Ahora, utilizando siempre WinSCP, configuramos el paquete motion. Vamos a la carpeta /etc/ y hacemos clic sobre el nombre de archivo "motion.conf" (**Fig. 19**).
Se abre un editor de texto con el contenido del archivo de configuración.
De toda la larga lista de parámetros de configuración analizamos y configuramos aquellos indispensables para el arranque inicial del paquete. Todos los otros parámetros podrán ser configurados posteriormente utilizando la interfaz web.
Empezamos por el parámetro:

```
; area_detect value
```

Se no especificamos nada, será analizada toda el área cubierta

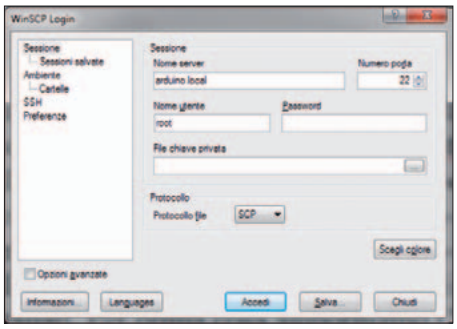


Fig. 14

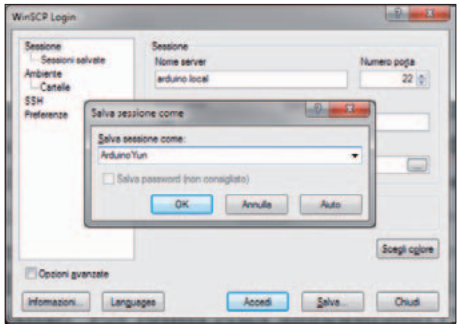


Fig. 15

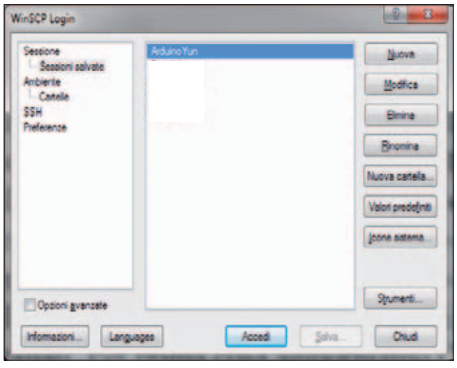


Fig. 16

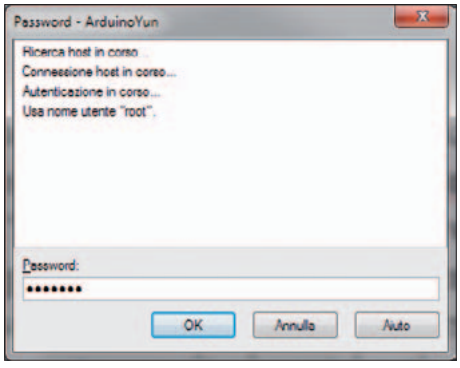


Fig. 17

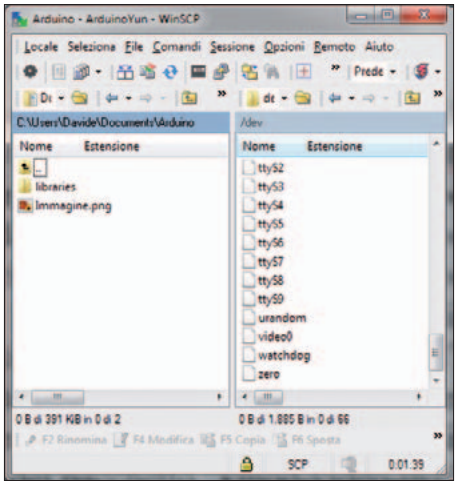


Fig. 18

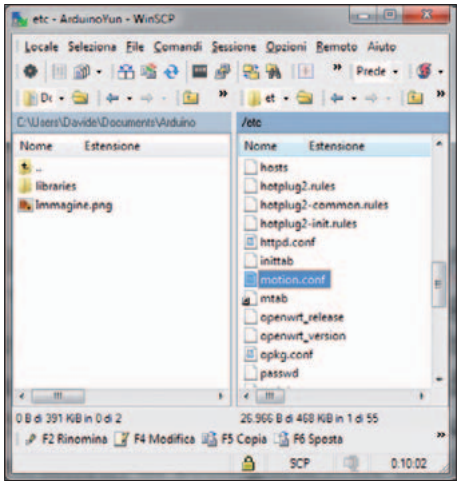
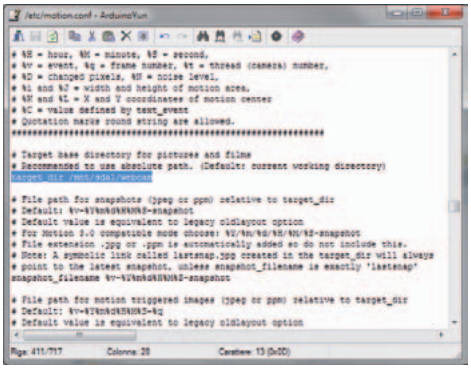


Fig. 19

Fig. 20

por la webcam para encontrar cualquier movimiento. Si queremos tener bajo control solamente una porción de la imagen, lo podemos hacer aplicando este parámetro. El área está dividida en nueve partes denominadas como en la **Fig. 20**.
Para activar el reconocimiento del movimiento en áreas determinadas basta incluir en el parámetro los números correspondientes a las zonas a tener bajo control, obviamente después de haber



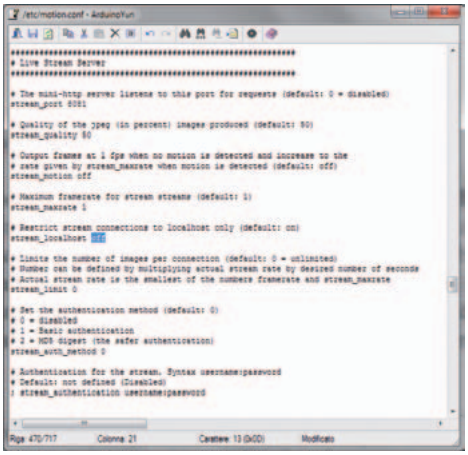


Fig. 22

quitado el ";" de comentario. Por ejemplo, para encontrar movimiento en la parte central de la imagen, de arriba a abajo, se utiliza la configuración:

```
area_detect 258
```

Por el momento dejamos las cosas como están de tal manera que toda el área encuadrada por la cámara sea analizada al terminar de reconocer un movimiento.
Vayamos sin embargo al parámetro de configuración:

Fig. 21

```
target_dir /usr/local/apache2/htdocs/cam1
```

Este parámetro contiene la carpeta predefinida en la que se memorizan todas las imágenes adquiridas por la cámara. La configuración predefinida prevé guardar solo las imágenes recogidas durante las detecciones de movimiento, de manera que se pueda reconocer al intruso, a posteriori. Si dejamos la carpeta predefinida, las imágenes se escribirán en la memoria NAND de Arduino Yún.
Debido a la reducida dimensión de la memoria, en pocos minutos las imágenes memorizadas saturaran todo el espacio disponible, poniendo en crisis también el funcionamiento general del sistema operativo Linino.
Menos mal que si ocurre, en general vasta cancelar los archivos y volver a lanzar todo. En el peor de los casos se restablece el sistema original con el pulsador de reset "WiFi reset button" manteniéndolo pulsado por más de 30 segundos. Para evitar cualquier inconveniente modificamos la configuración predefinida con la siguiente (**Fig. 21**):

```
target_dir /mnt/sda1/webcam
```

El parámetro:
`stream_localhost on`

permite limitar la visión del streaming de la webcam solo a los usuarios locales, no es nuestro caso ya que nosotros nos conectamos con el buscador de nuestro PC. Entonces habilitamos la conexión desde el exterior modificando el parámetro de configuración en la (**Fig. 22**):

```
stream_localhost off
```

Del mismo modo modificamos el parámetro que restringe a los usuarios locales el uso del instrumento de configuración vía web del server motion, modificándolo desde:

```
webcontrol_localhost on
```

en la (**Fig. 23**):

```
webcontrol_localhost off
```

La última configuración sirve para ejecutar el programa "allarme.py" cuando se observa un movimiento en el área controlada por la webcam (**Fig. 24**).

```
on_motion_detected "python /mnt/sda1/pgm/
allarme.py"
```

Ahora ocupémonos de preparar los programas para la gestión del evento que ocurre cuando se detecta un movimiento en la zona del campo de visión de la cámara configurado como zona "sensible". Configuramos el parámetro on_motion_detected de tal manera que se ejecute un programa python cuando viene cargado un

Fig. 23

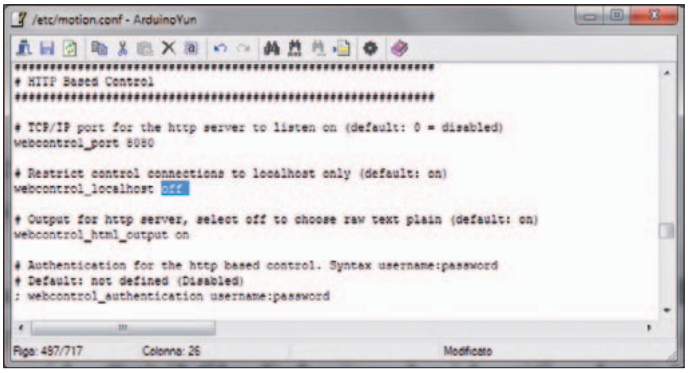
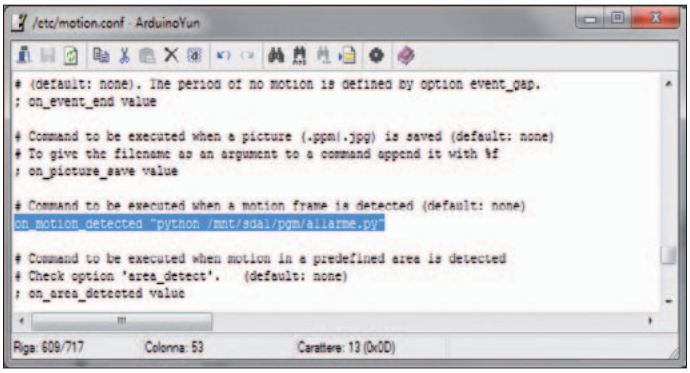


Fig. 24



Listado 1

```
#!/usr/bin/python

import os                                       #Importa la librería para ejecutar comandos GNU/Linux desde
pgm
import smtplib                                  #Importa la librería para enviare email
from email.mime.multipart import MIMEMultipart  #Importa módulos gestión email con adjuntos
from email.mime.text import MIMEText            #Importa módulos para adjuntos de tipo texto
from email.mime.image import MIMEImage          #Importa módulos para adjuntos imágenes

# Abre el archivo alarmas.txt en sobreescritura (w+) escribe el valore 1 y vuelve a cerrar
fileout = open("/mnt/sda1/pgm/allarme.txt", "w+")
fileout.write( "1");
fileout.close()

# Instrucciones para buscar el último fotograma memorizado en la carpeta /mnt/sda1/webcam
# a enviar como adjunto al email
carpeta = "/mnt/sda1/webcam"   #Importa la librería para ejecutar comandos GNU/Linux desde pgm

# Se posiciona sobre la carpeta de las imágenes
os.chdir(carpeta)

# Busca el fotograma más reciente
filelist = os.listdir(os.getcwd())
filelist = filter(lambda x: not os.path.isdir(x), filelist)
# Memoriza el nombre del fotograma más reciente para adjuntarlo porsteriormente
attachment = max(filelist, key=lambda x: os.stat(x).st_mtime)

# Ajusta los encabezados del mensaje de correo electrónico
to_addr = "<Direccion_mail_mail_de_destino>"
from_addr = "<Direccion_mail_mail_de_remitente>"
# Titulo del email
subject_header = "Alarmas"
# Texto del email
body = "Mensaje de alarmas"
# Instancia objetos mensaje como MIMEMultipart
msg = MIMEMultipart()
# Empieza a componer el mensaje por la cabecera
msg["To"] = to_addr
msg["From"] = from_addr
msg["Subject"] = subject_header
# Añade el texto del email
msgText = MIMEText('<plaintext>%s' % body, 'html')
msg.attach(msgText)
# Añade la imagen como adjunto
fp = open(attachment, 'rb')
img = MIMEImage(fp.read(), name = "imagen.jpg")
fp.close()
msg.attach(img)
# Inicia envió del email
# Ajuste servidor y puerto SMTP
emailRezi = smtplib.SMTP("<server_email>", <puerto_email>)
emailRezi.set_debuglevel(0)
# Ajuste usuario y password del usuario email
emailRezi.login("<usuario_mail>", "<password_mail>")
# Envia el email
emailRezi.sendmail(from_addr, to_addr, msg.as_string())
# Cierra la conexión con el servidor del email
emailRezi.quit()
```

evento.
Para escribir el programa utilizamos siempre WinSCP, vamos a la carpeta /mnt/sda1 y creamos una nueva carpeta "pgm". Para hacerlo hacemos clic con el botón derecho del ratón y seleccionamos "*Nuevo*">>"*Carpeta*". Escribimos el nombre "pgm" y guardamos, esperamos entonces que nos pidan el password y respondemos siempre con "arduino". Entramo en esta carpeta y creamos un archivo de nombre "allarme.py", siempre pulsando el botón derecho del ratón y eligiendo "*Nuevo*">>"*Archivo*". Lo guardamos. Después copiamos en el editor del programa las líneas del **Listado 1**. Los listados de los programas se pueden descargar de la web *www.nuevaelectronica.com.*
Guardamos y cerramos. Ahora abrimos otro archivo con nombre "allarme.txt", insertamos el valor "0" (sin comillas), guardamos y cerramos también este archivo.
Finalmente creamos un último programa de nombre "allarme_off.py", que será llamado por el sketch de Arduino para restablecer el contenido del archivo de alarmas. Copiamos las instruccio-

Listado 2

```
#!/usr/bin/python

fileout = open("/mnt/sda1/pgm/allarme.txt", "w+")
fileout.write( "0");
fileout.close()
```

nes del **Listado 2**.
En cuanto estemos, creamos también la carpeta "webcam" como subcarpeta de */mnt/sda1/* para acoger los fotogramas resultado del registro del movimiento.
El primer programa es el más complejo, sobre todo si consideramos la lógica "clásica" de Arduino. De hecho nos permite, además de "encender el semáforo" de detección de un movimiento - acción que se concreta en llevar al valor "1" el contenido del archivo "alarmas.txt" - envía un email a nuestra dirección de correo electrónico, con la notificación del evento y adjunto el ultimo fotograma capturado durante la detección del movimiento.
Una brevísima introducción al lenguaje Python antes de describir los programas. Python es un lenguaje interpretado y entonces los programas no tienen necesidad de ser compilados. También la estructura de los programas es muy simple. La especificación de los módulos y grupos de instrucciones se realiza en base al guion de las instrucciones. No tenéis llaves de apertura y cerrado de bloques de instrucciones y tampoco ";" de terminación de las instrucciones. Aun así es un lenguaje muy potente, que está disfrutando de una difusión siempre mayor y permite realizar aplicaciones con prestaciones de todo tipo. En cuanto a la lógica de los programas, hemos insertado comentarios casi en cada línea de código. Come podéis notar una gran simplificación viene del uso de librerías especializadas que ofrecen clases, objetos y métodos para el desarrollo de las funciones principales del programa. En nuestro caso hemos utilizado librerías

Listado 3

```
/*
Sketch gestión notificación movimientos de motion
*/
#include <FileIO.h>
#include <Process.h>

int led_rosso = 4; // LED de aviso de motion detection
int buttonPin = 5; // Pulsador reset LED rojo
char buttonState = 0;
String stato = "";

void setup() {
  // Setup Bridge llama bridge.py sobre Linino
  Bridge.begin();

  pinMode(led_rosso, OUTPUT); // Ajuste pin 5 como OUTPUT

  // Setup File IO llama files.py sobre Linino
  FileSystem.begin();
}

void loop() {

  // Instancia objeto archivo y Open del archivo en lectura
  File nome = FileSystem.open("/mnt/sda1/pgm/allarme.txt", FILE_READ);
  stato = "";    // Lectura de un carácter del archivo
  stato += (char)nome.read();
  nome.close();  // cierra el archivo allarme.txt

  if (stato == "1") {  // Si estado 1 y estado detectado un movimiento
     digitalWrite(led_rosso, HIGH);  // enciende el LED rojo de aviso
     Process zero_alarm;  // instancia objeto Process
     zero_alarm.begin("python");  // Comando que se pide ejecutar a Linino
     zero_alarm.addParameter("/mnt/sda1/pgm/allarme_off.py"); // Parámetros comando
     zero_alarm.run();  // petición de ejecución del comando
  }

  buttonState = digitalRead(buttonPin);
  if (buttonState == LOW) {          // Si se pulsa el pulsador
     digitalWrite(led_rosso, LOW);  // se apaga el LED rojo
  }
  delay(2000); // Retardo de 2 segundos por cada ciclo
}

Ahora podemos probarlo todo.
```

ya incluidas en la distribución Linino y por lo tanto inmediatamente disponibles.
Pasamos ahora al desarrollo del sketch para Arduino. El sketch utiliza la librería "Bridge" y los métodos disponibles para la gestión de los archivos sobre SDCard.
El esquema lógico del sketch, visible en el **Listado 3**, prevé de los siguientes pasos.
Al inicio del programa se incluyen las librerías que exponen los objetos que utilizaremos en el programa, o la librería Bridge para la gestión del "puente" de comunicación y la librería FileIO para la gestión de los archivos residentes en el archivo system de Linino. Definimos el pin 4 del LED y lo configuramos como salida. Definimos el pin 5 conectado al pulsador.
En la función *setup()* encontramos la instrucción *Bridge.begin()*, que

Fig. 25

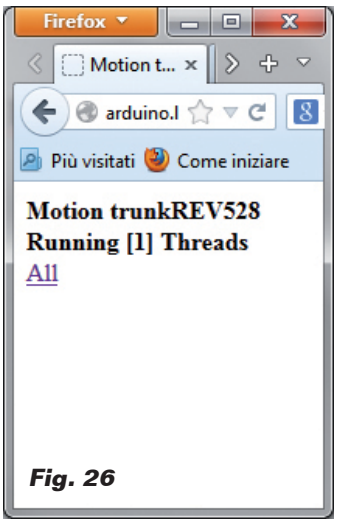


Fig. 26

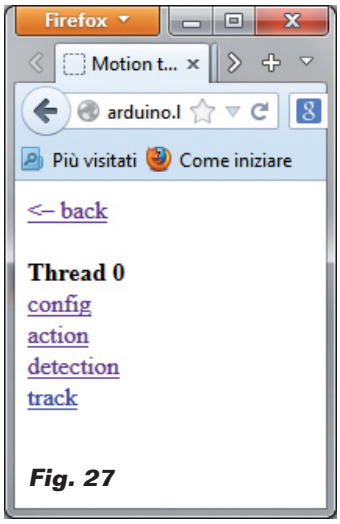


Fig. 27

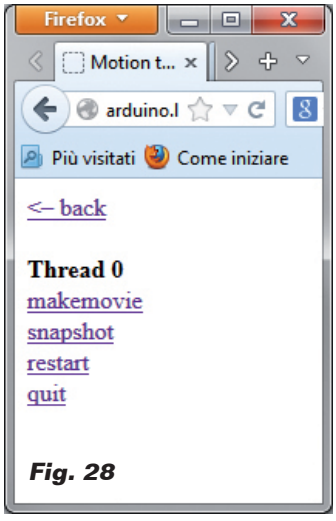


Fig. 28

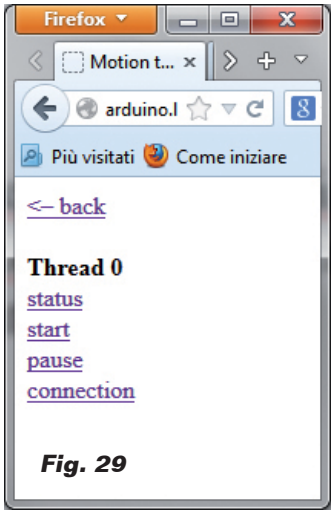


Fig. 29

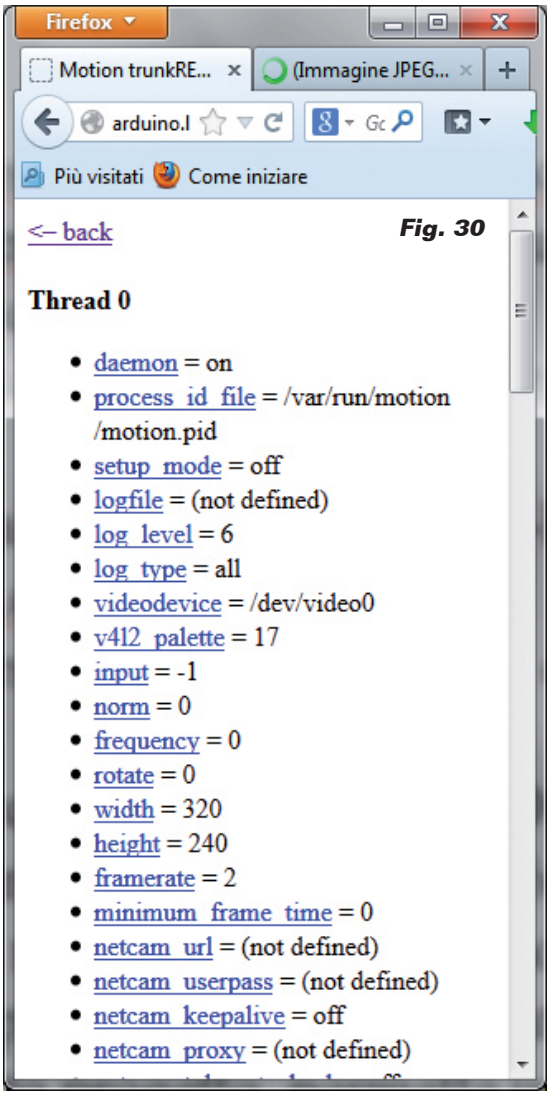


Fig. 30

inicializa el "puente" de comunicación entre las dos secciones de la tarjeta, de hecho llama a la carga en memoria, en entorno Linino, de los programas de gestión del puente de comunicación. La instrucción *FileSystem.begin()* llama en memoria en entorno Linino los programas que permiten a los sketch de Arduino de trabajar con el sistema de archivos de Linino. En el bloque *loop()* encontramos la lógica principal del programa. Abierto el archivo "allarme.txt" en lectura y se lee el contenido. Hay que subrayar que el método *read()* lee el contenido del archivo un byte a la vez. En nuestro caso no utilizamos un loop para la ejecución de todo el archivo, en cuanto hemos previsto una longitud del archivo de un solo byte. Cerramos enseguida el archivo para no interferir con una eventual petición de escritura por parte del server "motion". Continúa el test del contenido del archivo. Un valor "1" significa que ha sido derectado un movimiento del server "motion". En este caso se enciende el LED rojo y se procede a retornar al valor "0" el contenido del archivo "allarme.txt". En este caso la vía elegida para pasar a Linino es un comando que pida la ejecución, directamente en entorno Linino, del programa python "allarme_off.py", que escribe "0" en el archivo "allarme.txt". Los motivos son, por una parte, presentar en el mismo artículo una funcionalidad de las más importantes de Bridge y de la otra parte utilizar una modalidad de restablecimiento del archivo que interfiera lo menos posibles con el funcionamiento general de nuestra aplicación. Recordemos que el server "motion" está siempre atento para detectar un movimiento y notificarlo escribiendo el archivo "allarme.txt". Estas condiciones de potencial conflicto deben ser considerados durante el diseño de las aplicaciones embebidas como fuente de posibles errores ocasionales, difíciles de encontrar y solucionar. El método utilizado es crear un objeto "Process" que permite disponer de un comando para ejecutar en entorno Linino. La instrucción

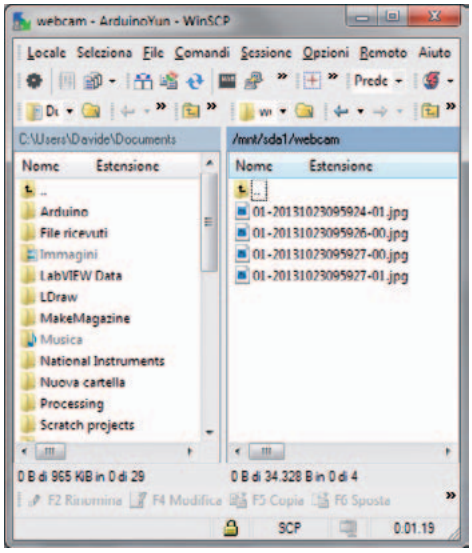


Fig. 31

```
zero_alarm.begin("python");
```

predispone el comando a ejecutar, es decir, la ejecución del interprete Python. A continuación añades los parámetros al comando, en nuestro caso el programa a ejecutar. La instrucción

```
zero_alarm.run()
```

ejecuta el comando en entorno Linino. Un retraso de dos segundos permite no saturar el "puente" de comunicación entre Arduino y Linino.
Ahora podemos probar todo.
Por la ventana del terminal de Putty, lanzamos el programa motion con el comando:

motion

Ahora abrimos un navegador de Internet y escribimos la dirección

http://arduino.localhost:8081

deberemos ver en el navegador la imagen grabada por la cámara (**Fig. 25**).
En una ventana separada escribimos la url:

http://arduino.localhost:8080

de modo que podamos acceder a la aplicación esencial de gestión de motion.

Obtenemos una página con el contenido de la **Fig. 26**.
Clicamos sobre "All" y obtenemos el menú de la **Fig. 27**.
Comenzamos por action que nos presenta el menú de la **Fig. 28**.
Con "quit" podemos interrumpir el funcionamiento del servidor, con "restart" lo inicializamos nuevamente, por ejemplo después de haber modificado algunos parámetros de configuración. "Makemovie" registra las imágenes grabadas por la cámara mientras que "Snapshop" guarda el fotograma actual.
El menú "detection" (**Fig. 29**) nos permite gestionar la actividad de reconocimiento del movimiento, mientras con "pause" y "start" podemos desactivarla y reactivarla.
Finalmente "config", con la función "list" (**Fig. 30**), nos permite personalizar los parámetros online, si los guardamos ("write"), las modificaciones se registraran en el archivo "motion.conf", que hemos modificado antes a mano. "get" y "set" nos permiten gestionar los parámetros individuales del archivo de configuración.
Ahora el test. Detectemos un movimiento ante la cámara, por ejemplo pasando delante con una mano, veremos encender el LED rojo y en la carpeta /mnt/sda1/webcam (Fig. 31) encontraremos una secuencia de imágenes que inmortalizan lo que ha sucedido. Abrimos nuestra bandeja de entrada de correo electrónico y encontraremos un mensaje de notificación con la imagen capturada adjunta (Fig. 32).
Ahora que hemos visto lo que ha sucedido podemos resetear todo accionando el pulsador sobre la breadboard, apagando el LED rojo. Este proyecto, permite una variedad enorme de experimentación, con la variación de los parámetros de motion y la gestión del evento mediante el sketch de Arduino. Podéis experimentar sin problemas: en el caso de situaciones "irreparables", aunque desde nuestra experiencia bastante difíciles de alcanzar, siempre es posible "empezar de nuevo" con el pulsador de reset.

(181085)

Fig. 32