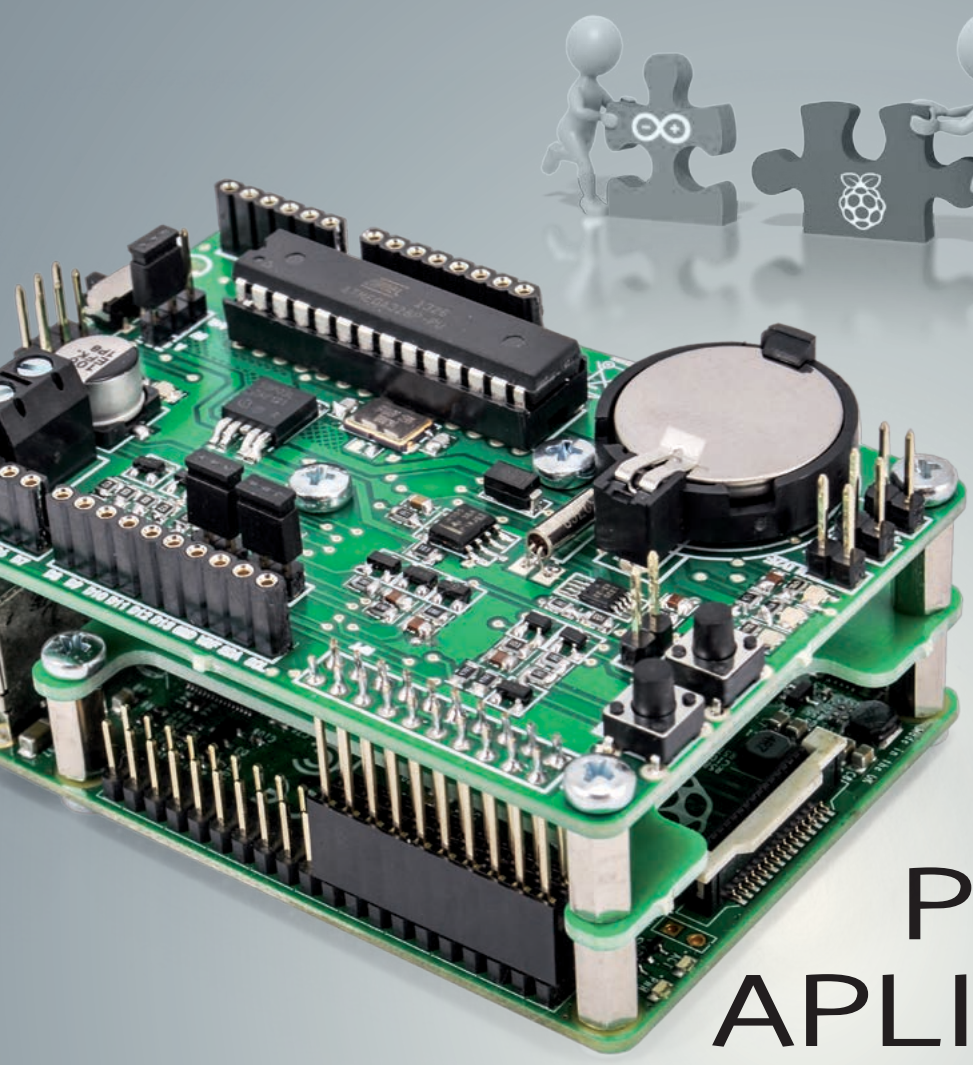


Practiquemos un poco con la tarjeta que hace de puente entre los mundos Arduino y Raspberry Pi, realizando un sofisticado sistema antintrusión.

RandA: PRIMERA APLICACIÓN

Ing. DANIELE DENARO

RandA es la tarjeta que hemos desarrollado para hacer cooperar Arduino y Raspberry Pi: se monta sobre esta última placa y abre un mundo de posibilidades que no existiría con las dos unidades tomadas por separado. Para imaginar los posibles usos basta un poco de fantasía, porque en ella están los elementos para utilizarla donde se utilizan tanto Arduino como Raspberry Pi. Antes de presentaros alguna aplicación compleja y completa en la que estamos pensando, para practicar un poco os proponemos algún ejemplo concreto, partiendo de una primera aplicación en la que se hace uso de un dispositivo muy extendido y quizás muy económico: la Webcam.

CONTROL AMBIENTAL VIDEO

Utilizando una Webcam estándar con interfaz USB y un servo de tipo micro (como por ejemplo el servo 7300-SERVO206 de Futura Elettronica, *www.nueva-electronica.com*), podemos construir una plataforma con Webcam giratoria de la que controlar el movimiento dentro de una área de 180 grados (**Fig. 1a** y **1b)**. Es necesario conectar la Webcam a Raspberry Pi, a través de un puerto USB, y los tres hilos del servo a RandA, teniendo en cuenta que el rojo (central) vaya sobre el pin correspondiente a 5V, el marrón va al pin GND y finalmente el amarillo a un pin de salida para el control: por ejemplo al D10. Para visualizar la imagen proporcionada por la cámara a través de la web podemos utilizar el software "motion" instalable sobre Raspberry Pi a través el comando:

sudo apt-get install motion

Pero antes es mejor actualizar el sistema a través del comando:

sudo apt-get update.

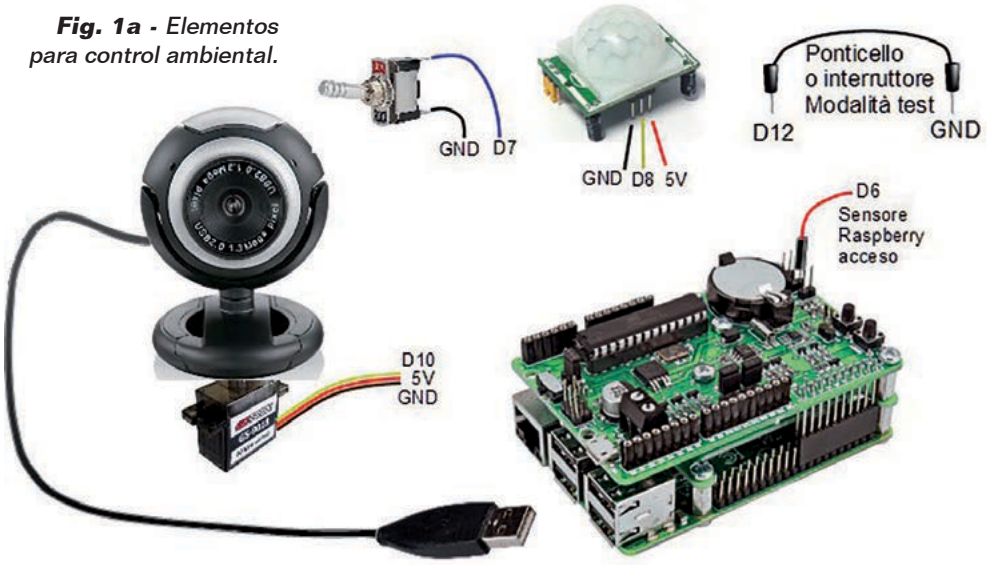


***Fig. 1a** - Elementos para control ambiental.*

***Fig. 1b** - Webcam con giro y PIR para detección intrusión.*

Este software tiene también una gestión de detección del movimiento integrada, aun así por esta vez utilizaremos, como detector de intrusión, un sensor PIR (por ejemplo 2846-PIRMOD): el motivo es que aprovecharemos la posibilidad que ofrece RandA de encender la Raspberry Pi, así que mantiene encendido solo Arduino, que tiene un consumo muy bajo. Para haceros una idea del escenario que se configura, en la **Tabla 1** están resumidos algunos datos de consumo; en ella vemos que con Arduino en power-down, el consumo se reduce a solo 7 mA (o 12 mA en configuración operativa para el control ambiental). Podremos entonces conectar el sensor PIR al pin D3, que corresponde a la interrupción externa número 1, para hacer despertar a Arduino, que a su vez activa la Raspberry Pi. Pero para no complicar demasiado el sketch, en esta aplicación nos contentaremos con consumir "bien" 28mA, en espera del evento "intrusión".

Resumiendo, solo en presencia de señalización por parte del sensor PIR, Arduino encenderá la Raspberry Pi, que ordenará disparar una foto, después la adquirirá y la mandara vía e-mail; con tal fin la Raspberry Pi permanecerá encendida para una eventual conexión vía web, permitiendo una visión continua y panorámica del entorno.

El control ambiental estará activado o desactivado accionando un interruptor conectado con Arduino (al pin D7) que permite la salida o entrada en el local bajo control.

Además otro pin (el pin D6) se usa para comprobar si la Raspberry Pi está encendida o no. Este pin está de hecho conectado al polo positivo del conector LD3, que está en paralelo a la alimentación; tened en cuanta que si el pin LD3 se utilizara para controlar un LED externo, la tensión en el polo positivo se reduciría a 1,2V y por tanto sería necesario leer el estado con una entrada analógica (por ejemplo A0).

Resuelta la parte mecánica y efectuadas las conexiones, se trata ahora de pasar al software.

En particular deberemos:

- configurar oportunamente el programa "motion";
- crear una página HTML en la cual visualizar el "stream" de la Webcam;
- insertar en la página los comandos para el giro;
- crear los script CGI para con-

***Tabla 1** - Consumos.*

Arduino Uno (alimentado por USB con reloj a 16MHz)	Funcionamiento normal. (Consumo debido a ATmega328P + circuitos de interfaz USB).	Alrededor de **48mA** (sin conexiones en los puertos).
	Funcionamiento en Sleep (Power Down Mode) despertable con interruptor externo (pin 2 o 3). Los circuitos de interfaz USB permanecen activos; el ATmega en realidad consume pocos microamperios.	Alrededor de **34mA** (debido casi completamente a la tarjeta).
RandA (con Raspberry Pi Plus y reloj del ATmega328P de 16MHz)	Funcionamiento normal Arduino+RaspberryPi Plus (Arduino comprende solo el ATmega328P, porque faltan los circuitos de interfaz USB).	Alrededor de **330mA.**
	Funcionamiento con Raspberry Pi apagada. (Solo Atmega328P pero con las conexiones a PIR y servo activas).	Alrededor de **28mA.** (23mA sin servo)
	Funcionamiento con Raspberry Pi apagada y Arduino en Sleep (Power Down Mode) despertable con interruptor externo. Esta vez la ATmega consume algún miliamperio también en power down a causa de los puertos conectados a Raspberry Pi.	**< 7mA** 3,2mA(ATmega) 3,7mA(tarjeta sin nada) 12mA con PIR y servo conectados

trolar Arduino;
- crear un sketch Arduino para detectar la señal del PIR, encender la Raspberry Pi y controlar el servo según los comandos recibidos por la Raspberry Pi en el puerto serie; el sketch debe además detectar la conmutación del interruptor de activación/desactivación.

EL PROGRAMA "MOTION"

El software "motion" es un potente instrumento (gratuito) de gestión de video con la finalidad de detectar el movimiento que ocurre frente a una Webcam. En la web *www.lavrsen.dk/foswiki/bin/view/Motion/WebHome* hay una completa documentación.

En realidad, "motion" tiene muchas otras funcionalidades: sobretodo comprende un servidor http dedicado al streaming del video de la Webcam; también se puede puede controlar la gestión de las características video también desde remoto, mediante otro puerto http (funcionalidad que nosotros no usaremos).

La actividad principal es la de detectar el movimiento sobre la base de la modificación con el tiempo de un cierto número de pixeles. Todos los parámetros, es decir, número de pixel necesarios para lanzar el evento, margen de error, posibles zonas del encuadre a controlar etc., son modificables actuando sobre un largo archivo de configuración de nombre "motion.conf".

En el caso realicen el que se produzca el evento, puede disparar una (o más) fotos, o registrar una grabación de duración preestablecida. Además, siempre en caso de un evento, puede lanzar un script; en la aplicación aquí descrita usaremos esta función para enviar un e-mail con foto adjunta.

En esencia, cuando nuestro sensor primario PIR detectara un movimiento, dirá a RandA que encienda la Raspberry Pi, ejecutará "motion". Después de un cierto e inevitable retardo debido al tiempo de startup de Raspberry Pi, "motion" estará listo para detectar en modo autónomo la actividad en el local bajo observación y entonces podrá lanzar la foto y enviar el e-mail. Está claro que, visto el objetivo de la aplicación, las decenas de segundos necesarios para el inicio de Raspberry Pi no tienen ninguna influencia sobre la captura del evento, que ciertamente no se termina en unos pocos segundos.

La Raspberry Pi permanece encendida y lista para recibir una eventual conexión http sobre la página HTML preparada a tal fin; la página permite la visualización del stream por el puerto del servidor de "motion" (puerto 8081, modificable). La página permite también enviar un comando de reset que restaura la situación de inicio, apagando la Raspberry Pi.

Para permitir el control de la visión de la Webcam, se ha preparado una modalidad "test Webcam", en la cual se enciende inmediatamente la Raspberry Pi cuando RandA detecta la conmutación del interruptor, y permanece encendida sin detectar el movimiento y sin disparar fotos. En este modo el servidor web está siempre disponible para la visión y el movimiento de la Webcam.

La modalidad "test" se activa, al iniciarse RandA, con la conexión a GND del pin D12.

El archivo de configuración estándar de "motion" se copia al hacer la instalación en el directo-

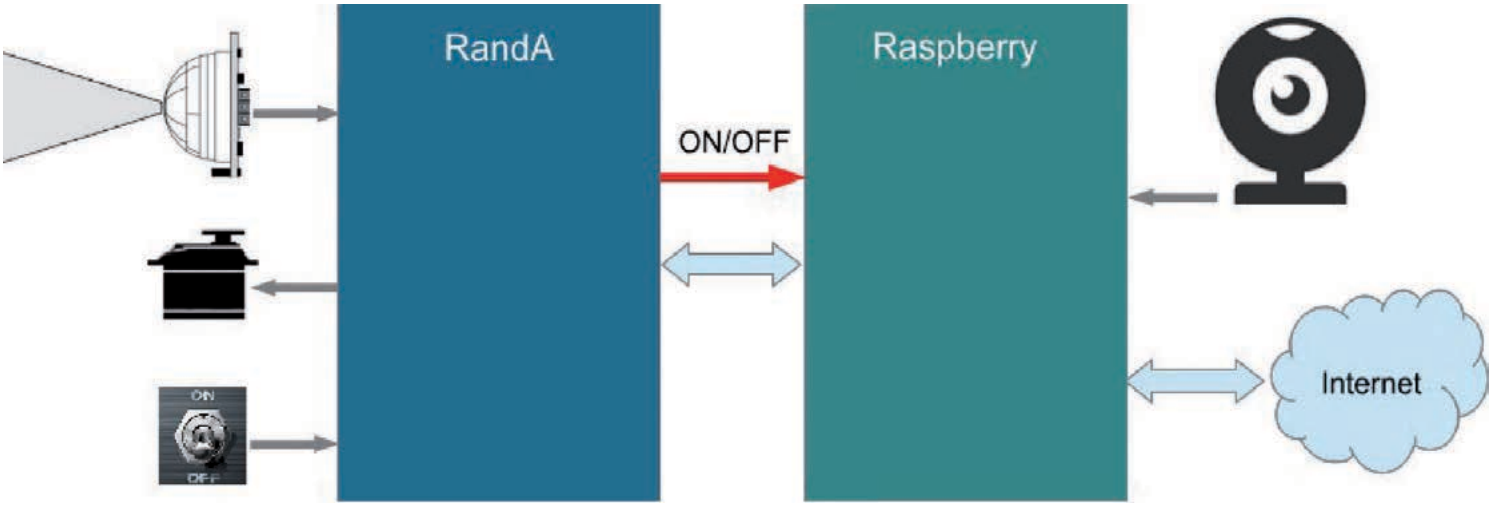


***Fig. 2** - Esquema general.*

Tabla 2

Variable	Valor original	Configuración para detecciones movimiento	Configuración para visión desde browser	Descripción
daemon	off	on	On	Parte en background.
process_id_file	/var/run/motion/motion.pid	/tmp/motion.pid	/tmp/motion.pid	Archivo que contiene el PID del proceso. Útil para cerrarlo.
framerate	2	24	24	Fotogramas al segundo por secuencia de foto y película.
output_normal	on	first	off	Lanza y guarda foto cuando detecta movimiento: off deshabilitada, first guarda la primera foto lanzada.
gap	60	10	60	Segundos para declarar cerrado el evento.
ffmpeg_cap_new	on	off	off	Filmati mpeg (disabilitati).
Webcam_maxrate	1	24	24	Fotogramas por segundo como Webcam.
Webcam_localhost	on	off	off	Visión solo local (no).
control_port	8080	0	0	Control remoto del programa (deshabilitado).
on_picture_save	*no activo (comentado)*	/home/pi/sendPicture.sh %f	no activo	Script activado cuando es lanzada una foto. %f contiene el nombre del archivo

rio "/etc/motion". Este archivo se adapta tanto para la actividad de control de movimiento como para la visión de la Webcam, razón por la cual haremos dos copias: una que posicionaremos en la carpeta "/home/pi" (carpeta del usuario pi) y otra que posicionaremos en la misma carpeta de los script CGI. La primera copia será utilizada en el caso de encendido de la Raspberry Pi para alarmas y tendrá el objetivo de configurar "motion" para disparar la foto y activar el script que la manda. La segunda será utilizada para activar la Webcam sin disparar foto.
Serán pocos los parámetros que se modificarám en el archivo de configuración, como se ha descrito en la **Tabla 2**.
De acuerdo a lo descrito en el archivo de configuración base, en caso de detección de movimiento se lanza una sola foto y está posicionada en la carpeta "/tmp/motion". El nombre del archivo contendrá una referencia a la marca de tiempo (timestamp) y está contenido en la variable "%f". Entre las distintas tipologías de eventos utilizables, nosotros usaremos solo aquella activada por el lanzamiento de la foto pasando al script también el nombre del archivo.
Hay que tener en cuanta que antes de mandar e-mail con el programa "SendMail" es necesario configurar el archivo "Mail.properties".
El script activado por el lanzamiento de la foto se llama "/home/pi/sendPicture.sh" y es visible en el **Listado 1**. El programa "motion" será lanzado con el comando:

motion -c *nombredelarchivodeconti-guracion*

Como se ha dicho, el programa "motion" será lanzado al encenderse la Raspberry Pi, que normalmente está apagada, y será alimentada para tomar la foto del movimiento y enviarla por e-mail. El sistema operativo Linux lanza, al arrancar, el script "/etc/rc.local", pero en vez de insertar el lanzamiento de "motion" al final de este archivo, aconsejamos modificar rc.local adjuntando una conexión a un eventual script posicionado en la carpeta del usuario; de tal manera podremos modificar el arranque de programas o script sin tener que ir a meter las manos sobre archivo de sistema y sin tener necesidad de autorización de "root".
Por ejemplo, como se muestra en el **Listado 2**.
En el **Listado 3** esta sin embargo descrito el script lanzado en el arranque.

Listado 1 – sendPicture.sh

```
#!/bin/bash
#Nombre de la foto a mandar en $1 (parametro pasado desde motion)
/home/pi/bin/SendMail mailto=pippo.pluto@gmail.com subject="Alarm!" attach=$1
#Realizado el trabajo cierra motion
sudo pkill motion
```

Listado 2 – a adjuntar al final de /etc/rc.local

```
# si el archivo de startup personalizado existe lo lanza como usuario pi
FSTARTUP="/home/pi/pistartup.sh"
if [ -x $FSTARTUP ];
then
  sudo -u pi /home/pi/pistartup.sh
fi
exit 0
```

PAGINA HTML DEL WEB SERVER

En esta aplicación el Servidor Web está activado solo como consecuencia de una alarma (excepto que en modo test); el encendido deberá estar habilitado solo a través de password, para impedir a extraños la posibilidad de mirar con la Webcam en nuestro local.
En Tomcat, para habilitar la protección es necesario intervenir sobre el archivo "/WEB-INF/web.xml" presente en la carpeta que contiene la aplicación. Para aislar la aplicación no usaremos la carpeta base "ROOT" del Servidor Web y creamosuna nueva carpeta en "webapps" que es la carpeta general de las aplicaciones web. De hecho, además de la aplicación base presente en ROOT, en webapps, se encuentran otras carpetas correspondientes cada una a una aplicación (que sea estática, con servlet o con uso de CGI). Pero cada carpeta debe contener las sub-carpetas META-INF y sobretodo WEB-INF con un archivo web.xml.
Creemos entonces una carpeta "ControlloAmbientale" en la cual creamos las dos subcarpetas "META-INF" y "WEB-INF". En "ControlloAmbientale" meteremos nuestra página HTML. Sin embargo en "WEB-INF" crearemos un archivo web.xml, como se ve en el **Listado 4**. A este punto deberemos crear la autorización para la aplicación interviniendo sobre el archivo "/home/apache-tomcat-7.0.47/conf/tomcat-users.xml" añadiendo la línea:

<user username="pippo" password="pluto" roles="Control" />

obviamente username y password han sido elegidos casualmente y debéis sustituirlos con aquellos que deseéis.
Finalmente crearemos la carpeta "cgi" en "WEB-INF" que contendrá los script bash de la aplicación. La página HTML (por ejemplo de nombre "WebCam.html") va colocada en la carpeta de la aplicación (ControlloAmbientale) y por tanto será referenciada como:

```
http:/...../ControlloAmbientale/WebCam.html
```

Listado 3 - pistartup.sh

```
#!/bin/bash
#exit 0          #para deshabilitar el script quitar el comentario
ser=/dev/ttyS0
stty -F $ser 9600
sleep 5
echo "M" > $ser
echo "Comando M" > /home/pi/start.log
  # retardo para dar tiempo a Arduino para responder
sleep 1
  # lee la respuesta de Arduino
read -r -t 2 replay < $ser
if [ ${#replay} -lt 4 ]; then
 echo "Ninguna respuesta!" >> /home/pi/start.log
 exit 1
fi
replay=${replay:0:4}
echo $replay >> /home/pi/start.log
if [ $replay = "TEST" ]; then
 echo "No motion" >> /home/pi/start.log
 exit 0                          #si en modo test sale
fi
if [ $replay = "CTRL" ]; then
 echo "Start motion!" >> /home/pi/start.log
  # de lo contrario cierra eventual proceso motion activo
 sudo pkill motion
  # y activa motion con detector de movimiento y envio foto
 motion -c /home/pi/motion-detect.conf
fi
```

Listado 4 - web.xml

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
                      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
  version="3.0"
  metadata-complete="true">

<description> Control ambiental con password</description>
<display-name>ControlloAmbientale</display-name>

<!-- Define el nombre de una regla de seguridad -->
<security-role>
  <description>ControlloAmbientale</description>
  <role-name>Control</role-name>
</security-role>

<!-- La regla de seguridad se entiende aplicada a la siguiente URL -->
<!-- El metodo de seguridad aplicado es aquel username-password basico -->
  <security-constraint>
    <web-resource-collection>
        <web-resource-name>ControlloAmbientale</web-resource-name>
            <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
            <role-name>Control</role-name>
    </auth-constraint>
  </security-constraint>

  <login-config>
        <auth-method>BASIC</auth-method>
    <realm-name>ControlloAmbientale</realm-name>
  </login-config>
<!-- -->
</web-app>
```

Fig. 3
Página HTML Webcam..

Para el encendido se pedirá la autorización a través del usuario y la password que hemos especificado. La página prevé el uso de un browser habilitado para HTML5 a causa del cursor (slide) y de la modalidad simplificada para visualizar el stream de la Webcam; os sugerimos Chrome, que presenta mejores gráficos para estos nuevos elementos de HTML5 respecto a Internet Explorer. La página sustancialmente contiene:

- el recuadro donde esta visualizado el stream de la Webcam;
- un cursor para mover la Webcam;
- tres pulsadores, que son uno para activar la recepción de la Webcam, uno para activar la modalidad auto-scan (giro continuo, tipo movimiento del limpiaparabrisas...) más un pulsador para hacer un eventual reset de las alarmas;
- un pulsador de refresh de señal de alarmas útiles en la modalidad test.

El recuadro para el stream es un simple campo imagen cuya fuente (source) hace referencia a otro servidor web con puerto 8081; de hecho el software "motion" puede crear también un servidor web propio para el direccionamiento del stream:

```
<div align="center">
<img id="vid" src="http://:8081/"
width="320" height="240"
style="background:#999; border:#CF3
groove"/> 
</div>
```

Naturalmente los pulsadores están controlados por funciones Javascript. La página HTML junto a todo el software de esta aplicación está contenida en el archivo comprimido "ControlloAmbientale.zip" *descargable de la web https://github.com/open-electronics/RandA/.*

Listado 5 - StartWCam.sh

```
##!/bin/bash

 # Deteccion de los parametros pasados de la llamada http
 # Estan presentes en la variable de ambiente QUERY_STRING
 # en el formato nombre1=val1&nombre2=val2...

qstring=$QUERY_STRING
 # Sustituye & con espacio para aislar las parejas par=val
qstring=${qstring//&/ }
 # Detecta las parejas par=val y las coloca en un array llamado par
read -r -a par <<< "$qstring"
ang=-1

 # Ciclo para cada elemento p del array par
for p in ${par[@]}; do
pn=${p%=*}                        # pn contiene el nombre
pv=${p#*=}                         # pv contiene el valor

 # parsing del comando
if [ $pn = "wcam" ]; then
 fstart=$pv
fi

done     # fin ciclo

 # accion

mconfig="/home/apache-tomcat-7.0.47/webapps/ControlloAmbientale/WEB-INF/cgi/motion.conf"
if [ $fstart = "OK" ]; then
 motion -c $mconfig            # lanza motion y responde a la llamada http
 echo "Content-type: text/html"
 echo ""
 echo "OK"
else
 pid=$(< /tmp/motion.pid)    # de lo contrario cierra motion y responde
 sudo kill $pid
 echo "Content-type: text/html"
 echo ""
 echo "NOK"
fi
```

FUNCIONES JAVASCRIPT PARA CONTROLAR LA WEBCAM

Las funciones actúan en modo AJAX y por tanto dialogan con el servidor web en background; en sustancia, lanzan script bash pasándoles parámetros (modalidad CGI). Para la activación de la Webcam basta un script bash sobre la Raspberry Pi, pero para el

movimiento de la cámara el script bash de la Raspberry Pi tendrá la tarea de mandar un comando a RandA a través del puerto serie. El cursor del giro llama a la función correspondiente solo cuando se suelta el pulsador del ratón, por tanto la posición de la Webcam se actualiza en modo no continuo.

SCRIPT CGI ACTIVADOS POR LAS FUNCIONES JAVASCRIPT

Para esta aplicación se ha decidido utilizar la modalidad CGI para no estar obligados a describir una aplicación con elementos de Java y Servlet que pertenecen al campo de las Web Application y no a todos estaréis familiarizados. Quien tiene practica en Java Web Application puede pasar fácilmente a implementar el dialogo AJAX a través de Servlet o JSP. Los script están conceptualmente divididos en dos partes: en la primera esta la detección de los parámetros enviados por la función Javascript, mientras en la segunda parte está la acción propiamente dicha. Los parámetros están comprendidos en la variable de entorno "QUERY_STRING" que el sistema operativo envía al script (es una actividad realizada por Tomcat). En la práctica se trata de marcar la línea para detectar le parejas *nombre=valor* separadas por el carácter '&'. Una vez realizada la función, si puede también responder por ejemplo a través de un "hecho ….". De hecho el stream de salida está interceptado por Tomcat y enviado vía http. Los script van colocados en la carpeta "/WEB-INF/cgi/" de la aplicación.
El primer script, "StartWCam.sh", lanza el programa "motion". El segundo, "RandAcmd.sh", ejecuta la interacción con RandA a través del puerto serie "/dev/ttyS0" que corresponde al puerto al cual RandA está conectado.

SKETCH PARA EL CONTROL AMBIENTAL

Como hemos dicho, el sketch tomará el control del sistema, encendiendo y apagando la Raspberry cuando sea necesario. Lo primero es, por tanto, de mover el puente de SW2 sobre la posición "Arduino siempre alimentado" (hacedlo con el sistema apagado). Los otros dos puentes JP1 y JP2 están cerrados como en configuración estándar. En el arranque, el sketch debe habilitar el pin D7 en modo "INPUT_PULLUP" para comprobar la posición del interruptor, el pin D8 en modo "INPUT" para recibir la señal del PIR y el pin D10 en modo "OUTPUT" para controlar el servo a través de la librería "Servo.h" que va por tanto incluida en el sketch. Además debe habilitar el pin D6 en modo "INPUT" para comprobar si la Raspberry Pi está encendida o apagada y el pin D12 en modo "INPUT_PULLUP" para comprobar si el modo actual es TEST o NORMAL. Finalmente debe ajustar el puerto serie a la velocidad predefinida (9.600) e inicializar el servo con el comando "attach". Del puerto serie llegaran los comandos de los script CGI para modificar la posición del servo. En la modalidad normal, en el "loop" continuo, el sketch deberá

1. leer el estado del interruptor para decidir si activar las otras funciones o mantenerse en standby (control desactivado);

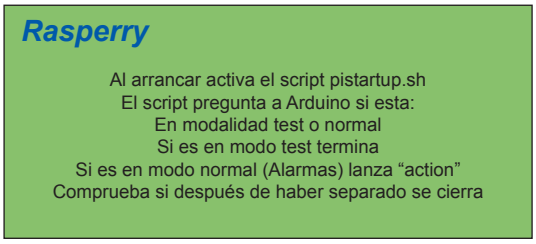
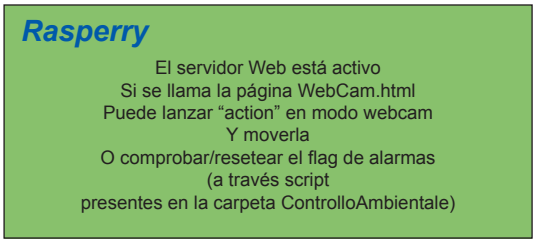


***Fig. 4** - Actividad sobre la Raspberry Pi.*



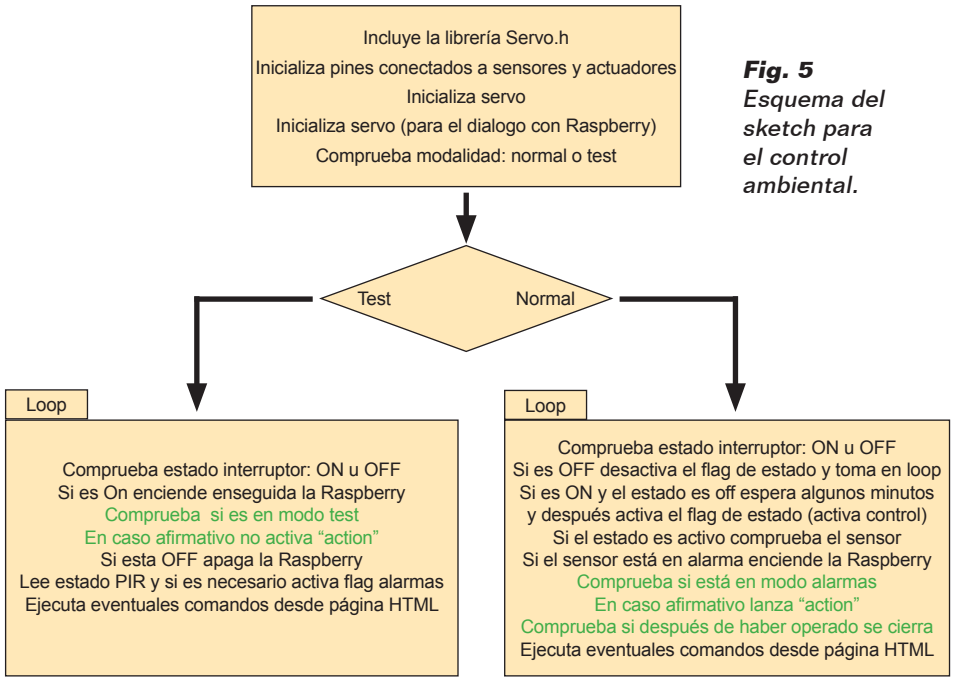


***Fig. 5** Esquema del sketch para el control ambiental.*

Listado 6 – RandAcmd.sh

```
#!/bin/bash
ser=/dev/ttyS0

 # Deteccion de los parametros pasados de la llamada http
 # Estan presentes en la variable de ambiente QUERY_STRING
 # en el formato nombre1=val1&nombre2=val2...

qstring=$QUERY_STRING
 # Sustituye & con espacio para aislar las parejas par=val
qstring=${qstring//&/ }
 # Detecta las parejas par=val y las coloca en un array llamado par
read -r -a par <<< "$qstring"

# Ciclo para cada elemento p del array par
for p in ${par[@]}; do
pn=${p%=*}                          # pn contiene el nombre
pv=${p#*=}                           # pv contiene el valor

 # parsing del comando y envio a Arduino
case $pn in
     QM)
        echo "M" > $ser
        ;;
     QA)
        echo "Q" > $ser
        ;;
     AN)
        echo "A"$pv > $ser
        ;;
     SC)
        echo "S"$pv > $ser
        ;;
     RA)
        echo "R" > $ser
        ;;
esac
done # fin del ciclo

sleep 0.2   # Retardo para dar tiempo a Arduino de responder
 # Lee la respuesta desde Arduino
read -r -t 2 replay < $ser

 # La repite detras como respuesta a la llamada http
 # pero primero debe cerrar el registro de la respuesta http
 # con una linea vacia.
 # El estandar output es automaticamente leido por el server Tomcat.
echo "Content-type: text/html"
echo ""
echo $replay   # respuesta http (a la funcion Javascript)
```

en el caso de pasar de estado inactivo a estado operativo, deberá esperar algún minuto antes de operar, para permitir la salida del local;

2. leer el estado del PIR para comprobar la llegada de alarmas; en caso afirmativo deberá proceder con el encendido de la Raspberry Pi y apenas esta esté lista, disparar una foto para adjuntarla al mail en que será enviada, dejarla encendida después hasta el eventual comando de stand-by enviado a través de la página HTML.

Siendo este un ejemplo demostrativo, hemos utilizado un simple interruptor para habilitar o deshabilitar el control ambiental. En una aplicación real deberemos mimetizar el interruptor o utilizar un teclado de seguridad con salida a relé, en la cual marcar un código de activación y desactivación. Podremos también gestionar la activación/desactivación por control remoto, para así deshabilitar el sistema antes de acceder al local y activarlo después de haber salido.

En la **Fig. 5** está representado el esquema en bloques del scketch; no publicamos el códogp fuente completo por razones de espacio, pero lo encontráis en el archivo descargable de nuestra web.

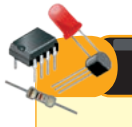
Para hacer más flexible el sistema, hemos insertado también una modalidad "test Webcam" que es activada poniendo a 0 (GND) el pin D12 con un puente o un interruptor.

Esta modalidad es activada al startup (o al reset) de RandA y hace así que la Raspberry Pi sea encendida enseguida sin considerar eventuales alarmas; así el sistema termina por trabajar como un servidor para la visualización y el movimiento de la Webcam.

CONCLUSIONES

Habéis visto cómo de fácil es realizar un control ambiental válido que nos permite también observar en streaming los efectos de la intrusión. Este sistema podría ser completado con una "llave" USB modem GSM/HSDPA para añadir el envío de SMS o para trabajar a través de la red móvil. En nuestra web está disponible el software completo, que debéis personalizar con vuestro password y dirección de correo electrónico.

(194083)



el MATERIAL

La tarjeta RandA está disponible ya montada y comprobada al precio de 39,00 Euros.

La placa Raspberry Pi Tipo B+ (cod. 8111284RS) se puede adquirir al precio de 36,00 Euros, mientras la Raspberry Pi 2 Tipo B (cod. 8326274RS) cuesta 46,00 Euros.

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