



Con Arduino Yún conectado a Internet y un visualizador de matriz de LED, visualizamos el texto enviado desde un smartphone a través de WiFi.

ARDU DISPLAY, CUANDO EL DISPLAY SE CONVIERTE INTERACTIVO

..... LUCA BELLAN

A finales de los años '90 del siglo pasado los displays de LED empezaron a sustituir o a integrar los clásicos letreros de los cines y de las actividades comerciales, no solo por el menor consumo, sino también por la posibilidad que ofrecen de componer escritos e imágenes a placer y de sustituirlas y cambiarlas en base a las exigencias del momento;

a esto hay que añadir la posibilidad de desplazar lo que es representado (text-scrolling), de manera que atraiga a los posibles clientes que pasan por la calle comunicando la programación, horarios de apertura, posibles promociones en los negocios, noticias y cualquier otra información útil. Si en los orígenes los displays promociona-



Fig. 1 - Time Square (New York) tras la llegada de los displays al LED.

les y los que vemos en las áreas públicas eran programados localmente y quizás reproducían los escritos y las imágenes enviadas desde un ordenador ubicado en el mismo sitio, hoy en día, en la época del IoT (Internet of Things), es posible controlar estos visualizadores remotamente, a través de WiFi o Internet, sustituyendo textos e imágenes o efectos sin tener que estar en el sitio. El proyecto aquí propuesto -que hemos bautizado ArduDisplay- entra en esta perspectiva, porque permitirá a las personas conectarse a una red WiFi con su propio dispositivo móvil (un smartphone, por ejemplo), escribir un mensaje a través de una página web y

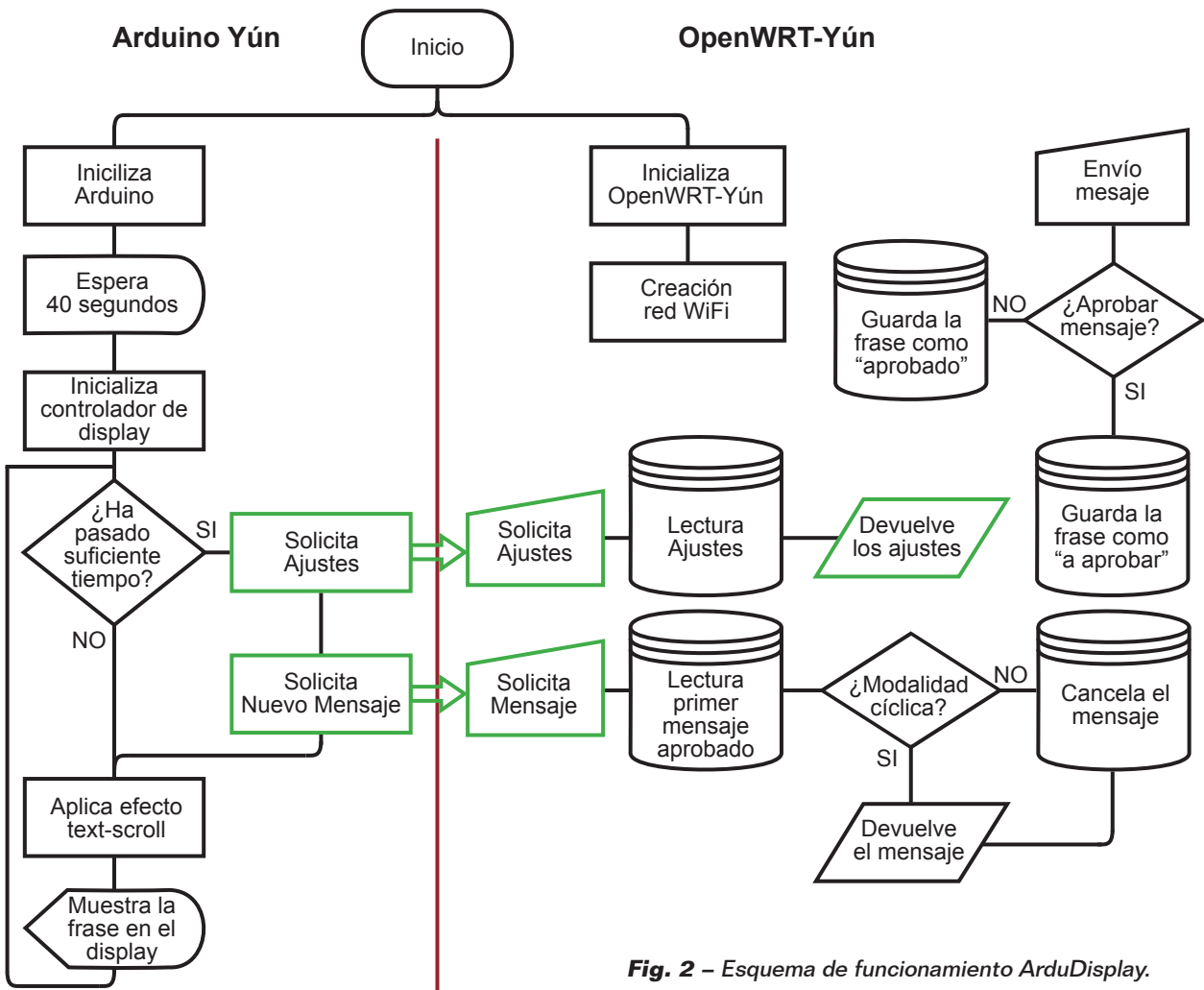


Fig. 2 - Esquema de funcionamiento ArduDisplay.

verlo visualizado sobre un display de LED que podremos posicionar en un stand en una feria, en el lugar de un concierto, en un local abierto al público, en un negocio, etc. Podemos usar el smartphone enfrente de nuestro negocio o del edificio donde está instalado el display y usarlo como “telemando” para escribir los mensajes.

Nuestra aplicación dispone de un panel de control (siempre disponible por la Red) que nos permitirá aprobar los mensajes recibidos desde el sistema remoto antes de mandarlos sobre el display, ajustar el tiempo de visualización de cada mensaje individual, la velocidad de desplazamiento del texto y elegir un mensaje predefinido para visualizar a falta de mensajes recibidos por el sistema.

Es también posible decidir no filtrar los mensajes, o sea enviarlos a la cola de visualización y visualizarlos sin tener que aprobarlos antes. Los mensajes recibidos por el sistema serán cancelados una vez mostrados sobre el display, pero podremos también elegir visualizarlos cíclicamente: desde el primero recibido hasta el último para después volver a empezar desde el primero.

NUESTRO SISTEMA

Para realizar ArduDisplay necesitaremos:

- Arduino Yún;
- de uno a cuatro display de LED Sure 3208 (depende de cuánto queramos que sea de largo nuestro display);
- un shield de conexión entre Arduino y los display, o jumper macho-macho, según el esquema de la Fig. 5;
- un microSD (de 512 MB o más).

Hemos elegido realizar este proyecto utilizando Arduino Yún porque esta tarjeta es una solución completa para interactuar vía

la web con los propios circuitos con la sencillez que caracteriza la plataforma Arduino. La conectividad Internet se obtiene a través de WiFi, gracias al módulo integrado en el Yún y a la distribución Linux llamada OpenWrt-Yun (que proviene de la más famosa OpenWrt) almacenada sobre microSD, que se ejecuta sobre el procesador de 400 MHz dedicado, provisto de 64 MB de RAM. Este equipo permite transformar nuestro Arduino en un servidor web equipado con PHP5 y base de datos MySQL.

El código fuente de Arduino se ocupará por tanto de preguntar por los mensajes a PHP (a través de la librería Bridge) y visualizarlos sobre el display, creando también el efecto de text-scrolling, mientras el Servidor Web permitirá a los usuarios enviar mensajes y a nosotros gestionarlos (aprobarlos o descartarlos).

Los mensajes y los ajustes (por ejemplo el tiempo de permanencia de la frase) serán guardados en una base de datos MySQL.

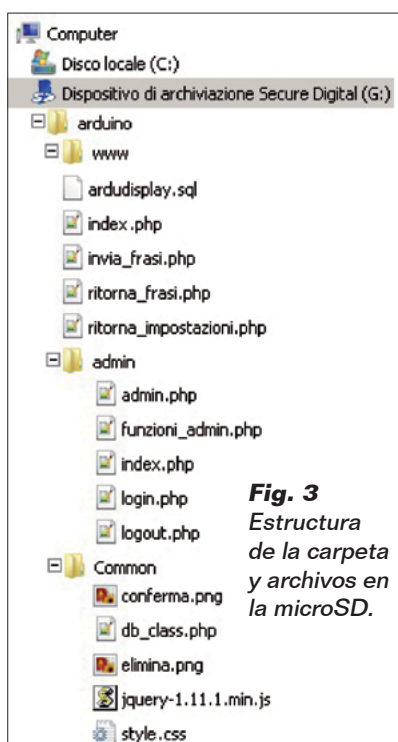
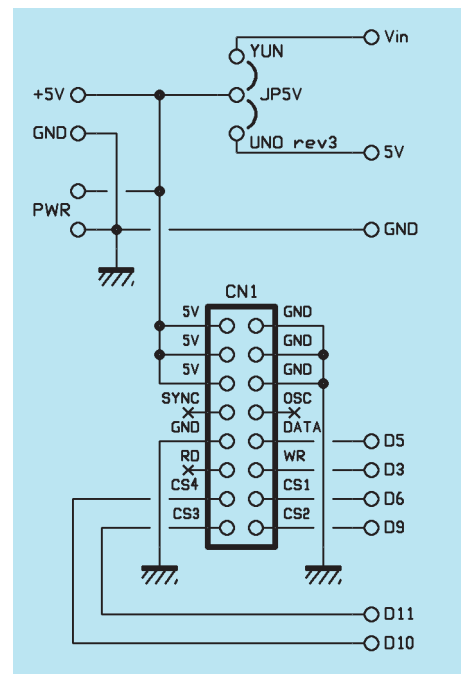


Fig. 3
Estructura de la carpeta y archivos en la microSD.



El archivo PHP y las páginas web serán memorizados sobre la tarjeta microSD, mientras la red WiFi a la cual se conectarán los usuarios será creada directamente por Arduino.

El display de matriz de LED es producido por la Sure Electronics, y es el modelo P432X8LEDMATRIX (adquirible en la tienda de Nueva Electrónica, www.nuevaelectronica.com): dispone de cuatro matrices 8x8 formadas cada una por 64 LED rojos de 3 mm de intensidad regulable y está basado en el controlador HT1632C de Holtek.

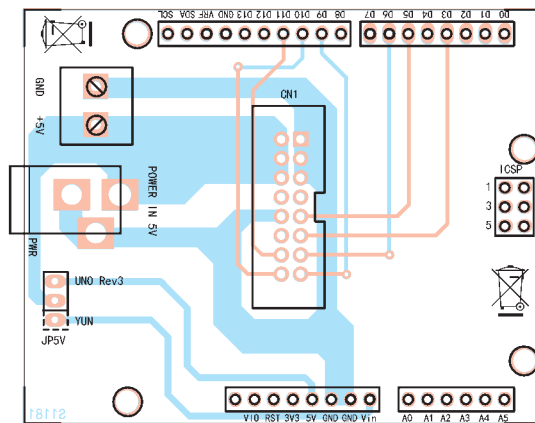
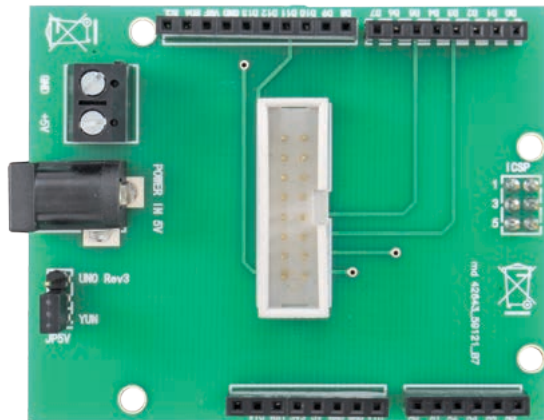
Podremos conectar hasta cuatro displays en cascada gracias al cable IDC de 16 pines previsto para ello; están además presentes dos terminales con tensión de salida 5 V, en el caso en el que la corriente entregada por nuestro Arduino no bastase para alimentar todos los displays. La interfaz de comunicación es un bus serie.

Nuestro proyecto utilizará, con el fin de simplificar, solo dos de estos displays, sin embargo cambiando los ajustes en la parte inicial del sketch, podéis tranquilamente realizar sistemas más extendidos, hasta de cuatro

[plano de MONTAJE]

Lista de materiales:

- Conector header de 8x2 pines para PCB
- Tira de 3 pines macho
- Tira de 6 pines macho/hembra
- Tira de 8 pines macho/hembra (2 pz.)
- Tira de 3x2 pines macho/hembra
- Tira de 10 pines macho/hembra
- Conector de alimentación
- Clema 2 polos para PCB
- Jumper
- Circuito impreso



módulos.

Cada display, con todos los LED encendidos a la máxima luminosidad, requiere una corriente de 0,36 A; ya que utilizaremos la potencia predefinida (cerca de la mitad) y no tendremos todos los LED encendidos para mostrar el texto, podremos alimentar todo con los pines 5V y GND de Arduino (alimentado a su vez por el conector Micro-USB).

Como hemos dicho antes, la microSD albergará el archivo PHP y las páginas web que permitirán a los usuarios enviar los mensajes, a nosotros de gestionarlos y a Arduino de recogerlos para después visualizarlos sobre el display. El conexionado entre el primer display de la serie y Arduino Yún se hará con un shield que permitirá insertar directamente el cable IDC

sobre ella: encontráis el esquema eléctrico y el plano de montaje en estas páginas. Colocando solo conectores y un jumper, el montaje del shield es simple hasta el punto que no necesita indicaciones particulares. Una vez que lo hayáis realizado, cerrar el puente JP5V entre la base central y el Yún.

CONFIGURACION DE ARDUINO YÚN

Veamos cómo preparar nuestra plataforma de manera que obtengamos como resultado final un Servidor Web Arduino capaz de gestionar paginas PHP, memorizar los datos en una base de datos MySQL y crear una red WiFi stand-alone que se llamara "ArduDisplay" y no requerirá insertar una contraseña para acceder.

Después, una vez conectados a la red "ArduDisplay", escribimos la dirección "http://scrivi.mi" en el buscador, seremos redireccionados a la página para el envío de los mensajes al display.

Como primera operación descargaremos desde nuestra web www.nuevaelectronica.com el archivo ZIP que contiene el sketch de Arduino, las librerías necesarias para el funcionamiento del display y los archivos a copiar en la microSD. Una vez descargado y descomprimido el archivo ZIP obtendremos, además del sketch de Arduino en la carpeta HT1632, una carpeta de nombre "arduino" que deberá ser copiada en el raíz de nuestra microSD: la carpeta contiene la instalación de la base de datos de ArduDisplay y todos los archivos PHP que analizaremos más adelante. Después insertaremos la microSD en la slot del Arduino Yún sea nuevo, os aconsejamos resetear completamente la distribución OpenWrt guardada en su interior: así serán canceladas posibles instalaciones o configuraciones que podrían interferir con nuestro proyecto.

Para resetear la OpenWrt-Yún es necesario activar el pulsador de reset WiFi (situado al lado del USB-A) durante al menos 30 segundos: Arduino reseteará todos los ajustes originales, como si fuera recién comprado; esta operación eliminara entre otras cosas todos los archivos instalados y los ajustes de red. Tal proceso os será útil si al realizar la aplicación alguna cosa fuera mal. Después de haber efectuado el reset, o después de haber encendido vuestro nuevo Arduino Yún, encontrareis una red WiFi de nombre "Arduino Yún-XXXXXXXXXXXX" (donde las X representan caracteres alfanuméricos que varían para

cada Arduino): conectémonos y escribamos "http://arduino.local" o "http://192.168.240.1" para acceder al panel de control, insertamos la contraseña (la predefinida es "arduino"), hacemos clic en "Configure" y elegimos una red WiFi preexistente a la cual conectarnos porque necesitaremos acceso a Internet; escribimos la contraseña en el caso en el cual sea solicitada y hacemos clic sobre el pulsador "Configure & Restart". Nuestro Arduino se reiniciará conectándose a la red WiFi establecida por nosotros; en este punto debemos encontrar la IP que le ha sido asignada por nuestro router (por ejemplo: 192.168.0.6) porque nos servirá para lanzar los comandos a través SSH en el próximo paso.

La dirección IP se puede consultar en la página de configuración de nuestro router (solamente accesible desde el buscador con la dirección 192.168.0.1 o 192.168.1.1) en la sección "Dispositivos conectados".

Los próximos pasos serán ejecutados todos desde línea de comandos; deberemos por tanto conectarnos en SSH a nuestro Arduino utilizando PuTTY (descargable gratuitamente desde <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html>).

Después de haberlo descargado y arrancado, insertamos en el campo "Host Name" la dirección IP de Arduino y hacemos clic sobre el pulsador "Open"; realizamos entonces el acceso con el usuario "root" y la contraseña que hemos establecido a nuestro Arduino (la misma que utilizamos para conectarnos al panel de control). Para instalar PHP5 lanzamos los siguientes dos comandos en sucesión:

```
opkg update
opkg install php5 php5-cgi
```

Editamos el archivo de configuración de uHTTPd (el web server integrado en OpenWrt-Yún) para incluir PHP:

```
nano /etc/config/uhttpd
```

Quitamos la marca de comentario de la línea:

```
list interpreter ".php=/usr/bin/php-cgi"
```

Adjuntamos la siguiente línea para hacer reconocer a uHTTPd los archivos con extensión ".php":

```
option index_page "index.php"
```

Cerramos y guardamos el archivo (CTRL+X, Y y Envío).

Finalmente reiniciamos el Web Server uHTTPd para hacerles leer los nuevos ajustes sobre el archivo de configuración:

```
/etc/init.d/uhttpd restart
```

Una vez instalado PHP5 y configurado uHTTPd, podemos pasar a instalar MySQL lanzando el siguiente comando:

```
opkg install libpthread libncurses
libreadline mysql-server
```

Abrimos el archivo de configuración de MySQL:

```
nano /etc/my.cnf
```

Asignamos a las variables "datadir" y "tmpdir" los siguientes valores:

```
datadir      = /srv/mysql
tmpdir       = /tmp
```

Cerramos el archivo guardando (CTRL+X, Y y Envío) y lanzamos los siguientes comandos en secuencia.

```
mkdir -p /srv/mysql
mysql_install_db --force
/etc/init.d/mysqld start
/etc/init.d/mysqld enable
mysqldadmin -u root password 'admin'
```

De esta manera habremos creado

un MySQL server, asociándoles el usuario "root" y la contraseña "admin"; ahora deberemos instalar el módulo MySQL para PHP; para hacerlo lanzamos este comando:

```
opkg install php5-mod-mysql
```

Abramos ahora el archivo de configuración de PHP, con el comando:

```
nano /etc/php.ini
```

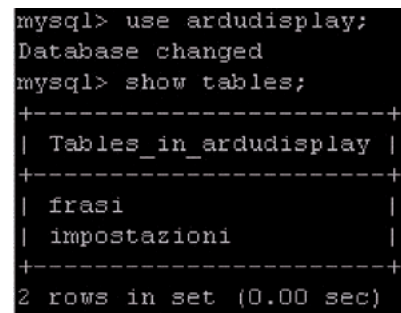
y quitamos la marca de comentario de la siguiente línea:

```
extension=mysql.so
```

Verificamos además que los parámetros del siguiente bloque de código estén compilados así:

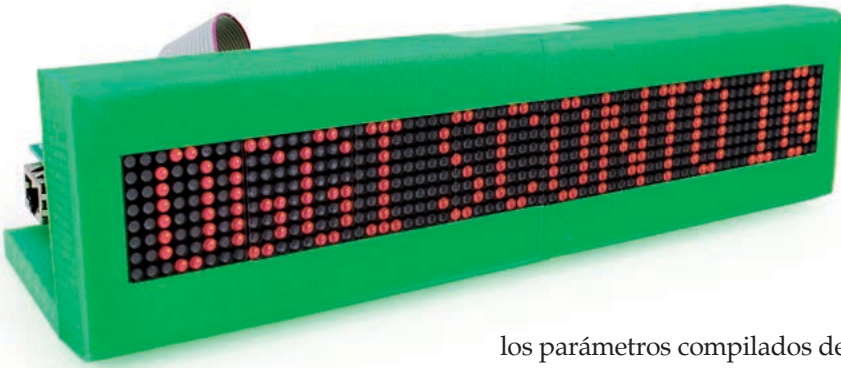
```
[MySQL]
mysql.allow_local_infile = On
mysql.allow_persistent = On
mysql.cache_size = 2000
mysql.max_persistent = -1
mysql.max_links = -1
mysql.default_port = 3306
mysql.default_socket = /tmp/run/
mysqld.sock
mysql.default_host = 127.0.0.1
mysql.default_user = root
mysql.default_password = admin
mysql.connect_timeout = 60
mysql.trace_mode = Off
```

Guardamos y cerramos (CTRL+X, Y y Envío).



```
mysql> use ardudisplay;
Database changed
mysql> show tables;
+-----+
| Tables_in_ardudisplay |
+-----+
| frasi                  |
| impostazioni          |
+-----+
2 rows in set (0.00 sec)
```

Fig. 4 – Tablas presentes en la base de datos ArduDisplay.



Después de haber creado el server MySQL debemos agregar nuestra base de datos y nuestras tablas. Movámonos por tanto a la carpeta donde encontraremos la instalación de nuestra base de datos:

```
cd /www/sd/
```

Abrimos la shell MySQL con el comando:

```
mysql -u root -p
```

e insertamos la contraseña de la base de datos. Sucesivamente lanzamos el siguiente comando SQL para crear la base de datos:

```
source arduisplay.sql;
```

Para comprobar que todo haya ido bien lanzamos estos dos comandos:

```
use arduisplay;
show tables;
```

Deberemos ver aparecer la tabla "frases" y la tabla "ajustes" como se muestra en la Fig. 4. Salimos por tanto de la shell de MySQL:

```
exit
```

Abrimos el archivo de configuración que será utilizado por nuestros archivos PHP para conectarse a la base de datos:

```
nano Common/db_class.php
```

Asegurémonos que este contenga

los parámetros compilados de esta manera:

```
$this -> mysql_server = "localhost";
$this -> mysql_username = "root";
$this -> mysql_pass = "admin";
$this -> database_name = "ardudisplay";
Guardamos y cerramos (CTRL+X, Y y Envío).
```

Ahora que hemos transformado nuestro Arduino en un web server con PHP y base de datos MySQL, debemos hacer de tal manera que, cuando el usuario escriba la dirección de Arduino sobre el buscador, no sea redireccionado al panel de control, y si a la página para enviar los mensajes; para hacer esto movámonos a la carpeta donde residen los archivos del web server con este comando:

```
cd /www/
```

Renombramos el archivo que nos redirecciona al panel de control con estos dos comandos:

```
cp index.html OLDindex.html
rm index.html
```

Creamos el archivo PHP que efectuará la redirección correcta:

```
nano index.php
```

Escribimos en su interior el siguiente código:

```
<?php
header("Location: sd/index.php ");
?>
```

Guardamos y cerramos (CTRL+X, Y y Envío). Después de haber vaciado la

cache del buscador, probando a insertar la dirección IP de Arduino seremos redireccionados a la página de envío de mensajes. Llegados a este punto, debemos hacer que Arduino cree su red WiFi stand-alone, por tanto tengamos pulsado la tecla de reset WiFi durante al menos 5 segundos (pero menos de 30, sino perderemos todo el trabajo realizado hasta ahora): así se volverá a crear la red inicial "Arduino Yún-XXXXXXXXXXXX". Conectémonos a esta red y vayamos al panel de control (<http://192.168.240.1/cgi-bin/luci/webpanel/homepage>), hagamos clic después sobre "advanced configuration panel (luces)", utilizando el menú principal movámonos en la sección "Network" y sucesivamente en la subsección "Hostnames".

En esta sección alcanzaremos una regla que transformara nuestra dirección IP en "http://scrivi.mi": hagamos clic sobre el pulsador Add, insertamos "scrivi.mi" en el campo Hostname y seleccionamos "--custom--" en el campo IP address, para después escribir "192.168.240.1"; finalmente hacemos clic en "Save & Apply". En este punto las direcciones que tendremos serán las siguientes:

- <http://scrivi.mi> para enviar un mensaje al display;
- <http://scrivi.mi/sd/admin> para gestionar los mensajes recibidos y otros ajustes (esta sección será analizada más adelante);
- <http://scrivi.mi/cgi-bin/luci/webpanel/homepage> para conectarnos al panel de control de Arduino.

La última operación que nos falta por efectuar es el cambio del SSID de la red WiFi de Arduino: conectémonos al panel de control avanzado (luces) utilizando la nueva dirección, deslicemos la página hasta la sección "Wireless" y hagamos clic sobre el nombre

de nuestra red.

Escribamos entonces "ArduDisplay" en el campo ESSID y hagamos clic sobre "Save & Apply", esperamos finalmente el reinicio de Arduino para después conectarnos a la red "ArduDisplay".

ANALISIS DE LOS ARCHIVOS PHP

Pasemos ahora a analizar las funciones de los archivos individuales PHP que hemos situado sobre la tarjeta microSD.

En la carpeta "www", que será la carpeta principal de nuestra web, encontramos el archivo index.php: este archivo será cargado justo después de haber escrito "http://scrivi.mi" y contiene la casilla de texto para enviar un mensaje al dispositivo.

El mensaje será recibido por el archivo envia_frase.php que se ocupa de escribirlo en la base de datos con un estado 0 (para aprobar) en el caso en el que hubiéramos establecido "Aprobar mensajes" a "Sí" en el panel de control, de otra manera 1 (aprobado), esto significa que nuestro mensaje será directamente puesto en la cola de visualización del display sin necesidad de ser aprobado.

Encontramos después el archivo ritorna_frase.php, que se ocupara de retornar de la base de datos una nueva frase cada vez que lo pida Arduino a través del objeto Process; en el caso en el que la modalidad "Ciclo de frases" esté desactivada, se extrae la primera frase en la cola de visualización (con estado 1), si no hay más se toma la frase predefinida que podrá ser cambiada en cada momento desde el panel de control.

Cada vez que será tomada una frase para mostrar, esta será cancelada de la base de datos inmediatamente.

Sin embargo si esta activada la modalidad "Ciclo de frases", la frase extraída no será cancelada,

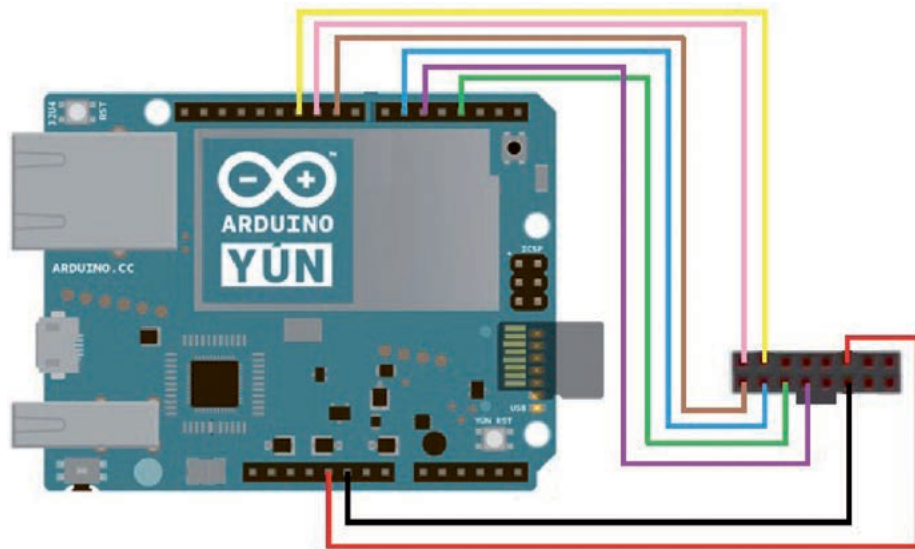


Fig. 5 – Conexión entre el primer display y Arduino mediante puentes de hilo: seguirlo si queréis ahorraros la construcción del shield.

ya que en esta modalidad serán visualizadas sobre el display todas las frases recibidas y, llegados a la última, el archivo PHP recomenzará a extraerlas desde la primera, incluyendo la frase predefinida.

El archivo ritorna_impostazioni.php es llamado por Arduino junto a ritorna_frase.php y se ocupa de transmitir a Arduino la duración (en segundos) de una frase sobre el display y la velocidad de scroll (también estos ajustables desde el panel de control que veremos más adelante).

Moviéndonos en la carpeta "admin", encontramos el archivo index.php que contiene una simple casilla de texto para efectuar el acceso al panel de control (a través del archivo login.php); la contraseña predefinida para acceder al panel de control es "admin", que se podrá modificar una vez efectuado el login.

La página admin.php nos permite efectuar múltiples operaciones: en el caso en que se active la modalidad "Aprueba frases" en el box de la izquierda veremos los mensajes para aprobar, en el box central los mensajes en cola de visualización, mientras en el box de la derecha

podemos modificar algunos ajustes de ArduDisplay:

- **Password admin:** es la contraseña para acceder al panel de control;
- **Durata frase display:** es el tiempo de permanencia (en segundos) de una frase sobre el display de LED;
- **Velocità scroll:** es la velocidad del efecto text-scrolling;
- **Approva frase:** ajustada a "Sí", las frases deberán ser aprobadas antes de ir a la cola de visualización;
- **Ciclo frase:** las frases, una vez enviadas al display, no son canceladas, se conservan ya que una vez que se muestra la última, Arduino comenzará otra vez desde el principio;
- **Frase default:** es la frase que será mostrada sobre el display en el caso no haya mensajes para mostrar

Haciendo clic sobre el pulsador "Logout" será llamado el archivo logout.php que nos permitirá salir del panel de control de ArduDisplay.

La carpeta "Common" sin embargo, contiene la librería javascript que nos permite efectuar

Listado 1

```
#include <Process.h>
#include <HT1632.h>
#include <font_8x4.h>
byte numero_display=2;
byte wr =3;
byte data = 5;
byte cs1 = 6;
byte cs2 = 9;
byte cs3 = 0;
byte cs4 = 0;
int larghezzaTesto;
unsigned char* fraseCorrente =(unsigned char*) "";
unsigned char* fraseVecchia =(unsigned char*) "";
int translazione=0;
byte tempoDisplay;
int tempoDelay;
String id_frase="-1";
int iterazioni;
int cont=0;
void setup() {
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);
  Bridge.begin();
  digitalWrite(13, LOW);
  switch(numero_display){
    case 1:
      HT1632.begin(cs1,wr,data);
      break;
    case 2:
      HT1632.begin(cs1,cs2,wr,data);
      break;
    case 3:
      HT1632.begin(cs1,cs2,cs3,wr,data);
      break;
    case 4:
      HT1632.begin(cs1,cs2,cs3,cs4,wr,data);
      break;
    default:
      break;
  }
  for(int i=0;i<numero_display;i++){
    HT1632.renderTarget(i);
    HT1632.clear();
    HT1632.render();
  }
  delay(40000);
}

void loop() {
  if(cont==0){
    for(int i=0;i<numero_display;i++){
      HT1632.renderTarget(i);
      HT1632.clear();
      HT1632.render();
    }
    caricaImpostazioni();
    cambiaFrase();
  }
  controllidiDisplay();
  cont=(cont+1)%iterazioni;
}

void controllidiDisplay(){
  for(int i=0;i<numero_display;i++){
    HT1632.renderTarget(i);
    HT1632.clear();
    HT1632.drawText(fraseCorrente, OUT_SIZE * (numero_display-i)-translazio-
  ne , 0, FONT_8X4, FONT_8X4_END, FONT_8X4_HEIGHT);
    HT1632.render();
  }
  translazione = (translazione+1)*(larghezzaTesto + OUT_SIZE * numero_di-
  splay);
  delay(tempoDelay);
}

void caricaImpostazioni(){
```

las llamadas a los archivos PHP (jQuery), la hoja de estilo CSS, las imágenes de los pulsadores para aprobar o cancelar un mensaje recibido y el archivo db_class.php que hemos analizado en el anterior párrafo y contiene la clase PHP y los datos para conectarnos a la base de datos MySQL.

EL SKETCH PARA ARDUINO

En el archivo ZIP que habéis descargado encontrareis también un archivo llamado "ArduDisplay_v1.ino" y una carpeta llamada "HT1632".

El display utilizado por nosotros requiere una librería, representada por la carpeta arriba citada. Para poderla utilizar es necesario copiar la carpeta entera en el directorio libraries de Arduino (normalmente situada en "Documents\Arduino\libraries"). Ahora podemos conectar nuestro Arduino al ordenador con el cable USB y abrir el archivo con extensión ".ino" en Arduino IDE.

Comprobad que el puerto COM y el modelo de Arduino indicados por el IDE sean correctos; si es así podéis proceder a compilar y cargar el código.

Pasemos ahora a analizar el código presente en el archivo haciendo referencia al **Listado 1**.

Lo primero que hace es incluir las librerías: la primera gestiona la comunicación entre los dos procesadores presentes en la tarjeta, mientras las otras dos son necesarias para el funcionamiento del display (la tercera, en particular, asocia a cada carácter su representación gráfica).

Encontramos después el número de los displays y la declaración de los pines necesarios para su control, seguida de la declaración de las variables:

- *longezzaTesto* es la anchura en pixeles de la frase a mostrar;
- *fraseCorrente* y *fraseVecchia* in-

dican respectivamente la frase recién recibida y la precedente, esto se utiliza para decidir si resetear o no la posición del texto al cambio del escrito (en el caso en el que la frase no cambie, la frase no mueve de nuevo al origen, dando así sensación de continuidad);

- *traslazione* es el número de pixeles que está desplazada actualmente la frase respecto al origen;
- *tempoDisplay* contiene la duración de la frase en segundos;
- *tempoDelay* indica el tiempo en milisegundos a esperar entre una ejecución del loop y la otra (un tiempo menor implica una mayor velocidad de desplazamiento);
- *id_frase* es el identificador de la base de datos de la frase mostrada, y es necesario para implementar la funcionalidad "ciclo frases";
- *iterazioni* indica el número de veces que el ciclo loop debe ser ejecutado antes de cambiar una frase; este depende de *tempoDisplay* y *tempoDelay* (por ejemplo, una frase debe ser mostrada durante 10 segundos con un tiempo entre un ciclo y el otro de 100 milisegundos, esto durara 100 ciclos);
- *cont* contiene el número iteraciones que han sido efectivamente ejecutadas.

En el setup encendemos un LED que se apagará cuando el puente entre los dos procesadores de los que está dotada la tarjeta haya sido establecido; después inicializamos la



Listado 1

```

Process p;
String temp;
p.begin("/usr/bin/php-cgi");
p.addParameter("-q");
p.addParameter("/mnt/sdal/arduino/www/ritorna_impostazioni.php");
p.run();
while (p.available()>0) {
    char c = p.read();
    if(c!='|'){
        temp += c;
    } else {
        tempoDisplay=temp.toInt();
        temp="";
    }
}
tempoDelay=temp.toInt();
iterazioni=tempoDisplay*1000/tempoDelay;
}

void cambiaFrase(){
    Process p;
    String temp="";
    p.begin("/usr/bin/php-cgi");
    p.addParameter("-q");
    p.addParameter("/mnt/sdal/arduino/www/ritorna_frasei.php");
    p.addParameter(id_frase);
    p.run();
    while (p.available()>0) {
        char c = p.read();
        if(c!='|'){
            temp += c;
        } else {
            id_frase=temp;
            temp="";
        }
    }
    fraseVecchia=fraseCorrente;
    unsigned char b[temp.length()+1];
    temp.getBytes(b,temp.length()+1);
    fraseCorrente = b;
    if (fraseCorrente!=fraseVecchia){
        translazione=0;
        larghezzaTesto = HT1632.getTextWidth(fraseCorrente, FONT_8X4_END,
        FONT_8X4_HEIGHT);
    }
}
}

```

pantalla de manera distinta según el número de displays conectados y lo "limpiamos" para después esperar 40 segundos para el arranque de la red WiFi y del Servidor Web de Arduino.

En el ciclo loop controlamos si la variable *cont* es igual a 0: este evento se comprueba solo al arranque del programa o cuando una frase ha sido mostrada para comprobar el número necesario de ciclos; en el caso en que sea verdadero, apagamos el display durante el tiempo necesario para cambiar la frase.

Son por tanto rellamadas las funciones *caricaImpostazione* y *cambiaFrase*.

Se pasa después a la gestión

verdadera y propia del display: la frase se renderiza sobre cada pantalla con una translación de datos (específica de cada display) y después se procede a incrementarla, restableciéndola en el caso en el que la frase haya recorrido todas las pantallas; finalmente se espera por un periodo de tiempo que depende de la velocidad de desplazamiento (delay).

La última instrucción del loop incrementa la variable *cont*, restableciéndola en el caso en el que hayan efectuado las iteraciones necesarias.

Las funciones *caricaImpostazione* y *cambiaFrase* llaman, utilizando un objeto de tipo *Process*, un archivo PHP presente sobre la OpenWrt-

Yún, y gestionan la salida que reciben en forma de flujo serie de caracteres.

La función *caricaImpostazione* recibirá un texto del tipo “tempoDisplay | tempoDelay” por tanto se ocupará de extraer de la cadena recibida los dos parámetros y de asignarlos, convirtiéndolos, a las correspondientes variables; *cambiaFrase* sin embargo recibe un stream del tipo “id_frase | mensaje”.

Mientras es posible asignar directamente el id de la frase a su variable, para hacer comprensible a las librerías del display el mensaje recibido es necesario convertirlo del tipo String a Byte con la función *getBytes*. Procedemos finalmente a controlar si la frase nueva es igual a la antigua, sino anulamos el vector de translación, y concluimos calculando la anchura en pixeles de la nueva frase. Si quisiéramos implementar un número de display distinto al utilizado en el ejemplo es suficien-

te modificar la variable *numero_display* y asignar el pin correcto a las variables *cs1*, *cs2*, *cs3* y *cs4*.

CONEXIONADO DE LOS DISPLAYS

Después de haber configurado Arduino y cargado el sketch, pasamos al conexionado entre los displays y la tarjeta; podemos efectuar la conexión en dos modos: la manera más rápida es sin duda utilizar el shield dedicado, que podría autoconstruir siguiendo el esquema eléctrico y el plano de montaje (y la correspondiente lista de materiales); la alternativa consiste en cablear el circuito conectando el primer display a Arduino utilizando el cable IDC mediante los cables apropiados. El esquema que debéis seguir para el cableado es el mostrado en la Fig. 5, recordando que los puentes deben realizar las conexiones entre Arduino Yún y el primer display, según lo marcado en la Tabla 1. Una vez conectado el primer display a Arduino, podemos conectar en cascada el segundo (y cualquier otro) utilizando los cables IDC.

Independientemente de la modalidad de conexión elegida, no nos olvidemos de establecer la dirección de cada display (de 1 a 4) actuando con los dip switch colocados detrás: la dirección es determinante para recomponer correctamente los mensajes sobre el display.

Tabla 1

PIN Arduino	PIN Display
3	WR
5	DATA
6	CS1
9	CS2
10	CS3
11	CS4
5V	+5V
GND	GND

EL PROYECTO TERMINADO

Llegados a este punto podremos colocar nuestro ArduDisplay, alimentarlo (por ejemplo a través de Micro-USB) y esperar casi un minuto para la inicialización del software y la creación de la red WiFi. Si todo ha ido bien, Arduino mostrará la frase predefinida al no encontrar otra para mostrar. Como se puede ver en la Fig. 6: podemos conectarnos con cualquier dispositivo a la red “ArduDisplay”, escribir sobre el buscador “<http://scrivi.mi>” y enviare un mensaje de prueba.

Ahora movámonos a la página “<http://scrivi.mi/sd/admin>”, realizamos el acceso con la contraseña predefinida “admin” y confirmamos el mensaje apenas recibido: después de pocos segundos deberemos verlo sobre el display.

(192073) ■



Fig. 6 - Test de Ardudisplay.

el MATERIAL

Todos los componentes utilizados en este proyecto son fácilmente localizables en el mercado. El diseño del circuito impreso, así como el archivo .stl para la impresión en 3D del contenedor y el sketch para Arduino, pueden descargarse desde www.nuevaelectronica.com. Cada matriz de LED cuesta 22,00 Euros (cod. 7719-P432X8LEDMATRIX) mientras la tarjeta Arduino Yún cuesta 62,90 Euros.

Precios IVA incluido sin gastos de envío.

Puede hacer su pedido en:

www.nuevaelectronica.com

pedidos@nuevaelectronica.com