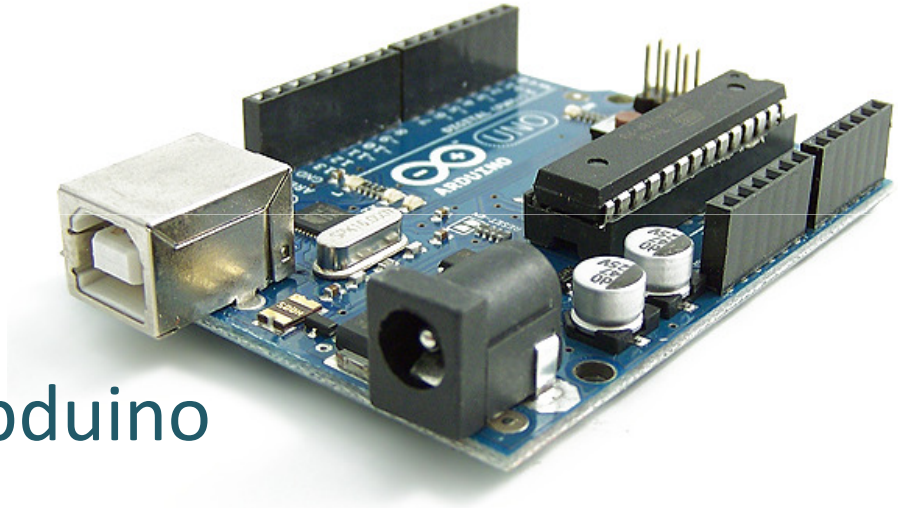
Aplicaciones Prácticas

con

Sistemas Arduino



Introducción a Webduino

CEP SEVILLA

IES Los Viveros

Curso 2011/2012

Ref: 11412FP23

:

Luis Modesto González

José Pujol Pérez

Webduino

- Librería que facilita la implementación de un servidor web con Arduino
- Permite que podamos interactuar con nuestro arduino a través de un navegador
- Será necesario crear una página web en html que deberá estar guardada en arduino.
- Enviando órdenes desde un formulario podremos ejecutar acciones
- Podemos añadir funcionalidades AJAX que permitan leer el estado de una entrada analógica en tiempo real (o casi)

Instalación

- Se descarga desde:
  - http://code.google.com/p/webduino/ o
  - https://github.com/sirleech/Webduino
- Creamos la carpeta WebServer dentro de librerías
- Copiamos los ejemplos dentro de examples
- Es necesaria la librería streaming, que descargamos desde http://arduiniana.org/libraries/streaming/
- Limitaciones:
  - 1 única conexión simultánea (el siguiente debe esperar)
  - No es posible SSL

Hola Mundo

- Abrimos el ejemplo “Hola Mundo”
- Ajustamos la dirección Mac
- Ajustamos la IP
- Abrimos el navegador con la ip de nuestro arduino

Cómo funciona un servidor web

- El cliente (mi PC) envía una petición al servidor en texto plano, que puede ser de tipo GET o POST
    - GET <shorturl> HTTP/1.1
    - Host: example.com
    - <other-headers>
    - <blankline>
    - POST <shorturl> HTTP/1.1
    - Host: example.com
    - <other-headers>
    - <blankline>
    - <url-encoded-content>

Cabeceras HTTP

```
http://www.ieslosviveros.es/

GET / HTTP/1.1
Host: www.ieslosviveros.es
User-Agent: Mozilla/5.0 (Windows NT 6.1; rv:8.0.1) Gecko/201
Accept: text/html,application/xhtml+xml,application/xml;q=(
Accept-Language: en,es-es;q=0.8,es;q=0.5,en-us;q=0.3
Accept-Encoding: gzip, deflate
Accept-Charset: UTF-8,*
Connection: keep-alive
Cookie: fontSize=80

HTTP/1.1 200 OK
Date: Sat, 03 Mar 2012 20:23:00 GMT
Server: Apache/2.2.9 (Win32) mod_ssl/2.2.9 OpenSSL/0.9.8h P
X-Powered-By: PHP/5.2.5
Set-Cookie: PHPSESSID=ear6l910fh9ii0h1gvdgbn3086; path=/
```

Como funciona un servidor web

- El servidor, contesta enviando un encabezado y un contenido.
- En el encabezado viaja una señal de estado
  - 200 OK
  - Content-Type: text/html
  - <other-headers>
  - <blank-line>
  - <HTML web page>

TIPOS DE APLICACIONES

- **Unidireccionales**. Arduino nos presenta el estado de algún sensor. Para refrescar, debemos hacer una recarga dela página.
- **Interactivas**. Arduino nos presente un menú, en el que tenemos algún elemento para ejecutar acciones, por ejemplo, un botón para encender o apagar.
- **Con autorefresco**. La página que nos presenta arduino, se refresca de forma automática cada cierto tiempo, así podremos detectar cuando ocurra algo.

HTML

- Las páginas que visualiza un navegador, deben estar escritas en HTML
- HTML, permite incrustar enlaces, por lo que podemos hacer que una página web servida por arduino, presente imágenes o ficheros alojados en otros servidores.

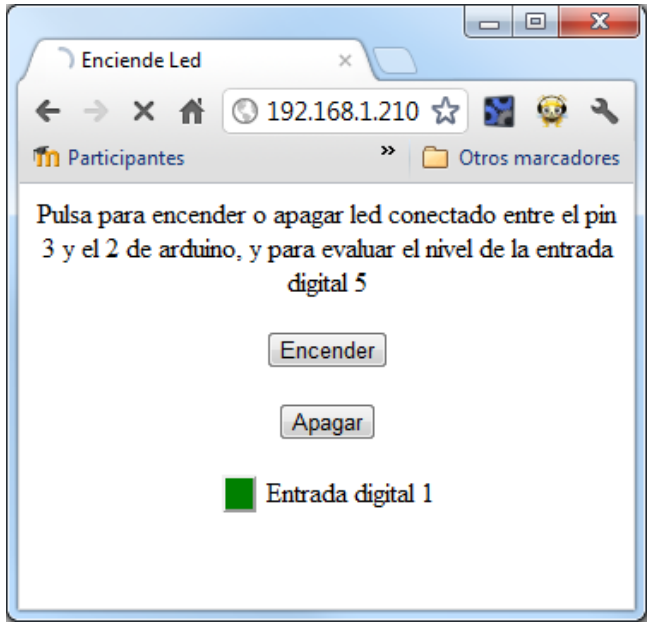
```
<html>
  <head><title>Hello, World</title></head>
  <body>
  <img src="http://www.cepsevilla.es/templates/mineown/images/header21.jpg”>
  <h1>Hello, World!</h1>
  </body>
</html>
```

Limitaciones de arduino

- Arduino tiene 2k de ram y 32k de Flash.
- Una aplicación web puede requerir bastante texto .
- Si escribimos:
    - server.print(" Hello! ");
    - El texto se guarda en RAM
- Si escribimos
    - P(message) = "<h1>Webduino</h1>";
    - server.printP(message);
    - El texto se guarda en Flash
- La librería Streaming, permite
    - server << "You have " << n << " bananas";
    - Frente a:
        - server.print("You have ");
        - server.print(n);
        - Server.print(" bananas";);

Formularios HTML

- Permiten añadir interactividad.
- Necesitaremos un editor HTML para diseñar el formulario y luego moverlo a arduino.
- Se admiten envío de parámetros por POST y por GET(desde la propia url)



Páginas con autorefresco

- Con un encabezado del tipo:
  - <META HTTP-EQUIV="Refresh" CONTENT="0.5">
  - Conseguimos que la página se refresque 2 veces cada segundo
- Podemos incluir scripts de javascript que autorefresquen la página cada x segundos.
- Podemos incluir librerías AJAX refresquen sólo los datos, sin recargar toda la página.

Programación con Webduino

- Tendremos que definir una función que será la que se encargue de tratar las peticiones web.

```
#define PREFIX "/carpeta"  /// ó ""
WebServer webserver(PREFIX, 80);
void tratamiento(WebServer &server, WebServer::ConnectionType type, char
*url_tail, bool tail_com)
{//Server: Objeto que representa al servidor
//ConnectionType tipo de petición; INVALID, GET, HEAD, POST, PUT, DELETE, PATCH
//url_tail : Parte de la url que no coincide con carpetas o ficheros registrados
//tail_complete:es true, si toda la url entró en el buffer, flase en caso contrario
//Tratamiento
}
void setup()
{
  Ethernet.begin(mac, ip);
   webserver.setDefaultCommand(&tratamiento);
}
void loop(){
  webserver.processConnection();
}
```

Ejemplo 1: Enciende Led.

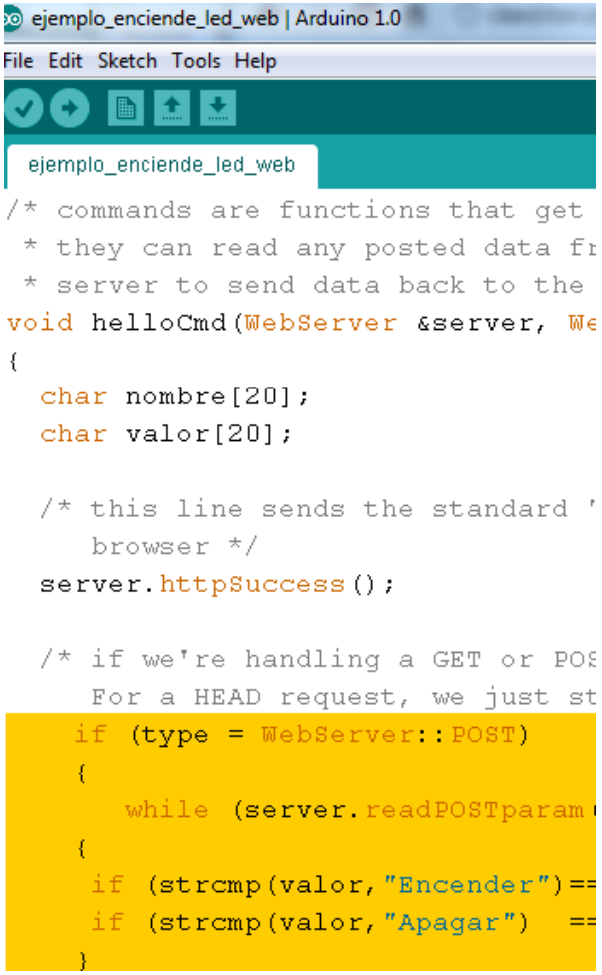
- Creamos la página principal con un editor html
- Abrimos el ejemplo “Hello world” y cambiamos el texto a mostrar por el del formulario

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<title>Enciende Led</title>
</head>
<body>
<center>
Pulsa para encender o apagar led en pin 3 de arduino
<form method="POST" action="">
		<p><input type="submit" value="Encender" name="en"></p>
		<p><input type="submit" value="Apagar" name="ap"></p>
</form>
</center>
</body>
</html>
```

Ejemplo 1: Enciende Led.

- Tenemos que procesar las solicitudes POST y ver si existe una parámetro Encender o Apagar, que es el “value” del botón pulsado.
- El código lo colocamos dentro del procesamiento del comando

```
if (type = WebServer::POST)
    {
       while (server.readPOSTparam(nombre,10,valor,10))
    {
     if (strcmp(valor,"Encender")==0) digitalWrite(3,HIGH);
     if (strcmp(valor,"Apagar")  ==0) digitalWrite(3,LOW);
    }
   }
```



Ejemplo 2: Lee entrada digital.

- Modificamos la página anterior ya añadimos un botón que cambiará a verde cuando la entrada esté alta.
- Añadimos el encabezado <META HTTP-EQUIV="Refresh" CONTENT="1">, para que se refresque cada segundo

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<META HTTP-EQUIV="Refresh" CONTENT="1">
<title>Enciende Led</title>
</head>
<body>
<center>
Pulsa para encender o apagar
<form method="POST" action="">
        <p><input type="submit" value="Encender" name="en"></p>
        <p><input type="submit" value="Apagar" name="ap"> </p>
        <p><input type="button" value=" " name="B1" STYLE="background-color:green" > Entrada digital 1</p>
</form>
</center>
</body>
</html>
```

Ejemplo 2: Lee entrada digital.

- Tenemos que insertar de forma condicional, el estilo de color del botón en función del valor de la entrada digital.
- Dividimos el mensaje en 2 trozos, e insertamos en medio el color del botón

```
P(helloMsg1) = "<html><head>"
"<meta http-equiv=\"Content-Type\" content=\"text/html; charset=windows-1252\">"
"<META HTTP-EQUIV=\"Refresh\" CONTENT=\"1\">"
"<title>Enciende Led</title>""</head><body><center>"
"Pulsa para encender o apagar led conectado entre el pin 3 y el 2 de arduino, y para evaluar el nivel de l
a entrada digital 5""<form method=\"POST\" action=\"\">      "
"       <p><input type=\"submit\" value=\"Encender\" name=\"en\"></p>"
"       <p><input type=\"submit\" value=\"Apagar\" name=\"ap\"></p>"
"<p><input type=\"button\" value=\" \" name=\"B1\" ";
P(helloMsg2) ="> Entrada digital 1</p> </form></center></body></html>";
```

```
char color[]="STYLE=\"background-color:green\"";
if (digitalRead(5)==0)strcpy(color," ");
server.printP( helloMsg1);
server.print(color);
server.printP( helloMsg2);
```