ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Proyecto Fin de Carrera

[Plataforma móvil para Aula Virtual basada en wrt54gl-openwrt-sdmod]

AUTOR: D. Fernando Sánchez Navarro
DIRECTOR: Dr. Francesc Burrull i Mestres
Mayo / 2012

Autor	D. Fernando Sánchez Navarro
E-mail del autor	fersanchna@gmail.com
Director	Dr. Francesc Burrull i Mestres
E-mail del Director	francesc.burrull@upct.es
Codirector(es)	
Título del PFC	Plataforma móvil para Aula Virtual basada en wrt54gl-openwrt-sdmod
Descriptores	

Resumen

Como consecuencia del desarrollo de las TIC la educación no presencial crece en importancia día a día a nivel mundial. En concreto, en el entorno universitario la educación a distancia es cada vez más utilizada. Si nos centramos en el caso de la UPCT, disponemos de la plataforma "Aula Virtual", basada en la tecnología Moodle, plataforma de eLearning de código abierto.

Por lo que respecta a los países subdesarrollados, básicamente el problema es el acceso a Internet, que suele ser prohibitivo para la gran mayoría de la población.

Para este segmento de población existen iniciativas para dotar de ordenadores portátiles con wifi y algunos contenidos educativos, pero el problema reside en que no tienen acceso a internet.

El objetivo de éste PFC es que los docentes pueden subir su material a un blog y los alumnos acceder a éste de manera local en institutos o universidades sin conexión a internet.

Titulación	Ingeniero Técnico de Telecomunicación, Especialidad en Telemática
Departamento	Tecnologías de la Información y las Comunicaciones
Fecha de Presentación	

A mis abuelos Pedro y Carmen
y a mis padres Fernando y Charo,
gracias a vuestra fe y a vuestro apoyo
he podido llegar hasta aquí.

ÍNDICE

1 INTRODUCCIÓN

No cabe duda que como consecuencia del desarrollo de las TIC la educación no presencial crece en importancia día a día a nivel mundial. En concreto, en el entorno universitario la educación a distancia o eLearning es cada vez más utilizada. Si nos centramos en el caso de la UPCT, disponemos de la plataforma "Aula Virtual", utilizada por miles de alumnos y profesores, basada en la tecnología Moodle, plataforma de eLearning de código abierto.

Por lo que respecta a los países subdesarrollados, el escenario cambia un poco. Básicamente el problema es el acceso a Internet, que suele ser prohibitivo para la gran mayoría de la población (con sueldos de aproximadamente 0,30€/hora).
Para este segmento de población (la mayoría del planeta) existen iniciativas para dotar de ordenadores portátiles con wifi (http://laptop.org) y algunos contenidos educativos. El problema suele estar no en disponer de ordenador, sino en acceder a información.

Quizás este PFC cabe dentro de estas iniciativas, ya que lo que se propone es utilizar una plataforma barata (aprox. 100€ minorista): Desde la capa hardware a las aplicaciones sería: el router wifi Linksys wrt54gl, una tarjeta SD de 2 Gb para contenidos, con un sistema operativo linux de código abierto, el openwrt y finalmente un servidor con soporte para wordpress con el cual los docentes puedan subir su material al blog y los alumnos acceder a éste de manera local (en institutos o universidades sin conexión a internet).

2 OBJETIVOS

El objetivo final del proyecto es integrar en un mismo producto una serie de aplicaciones para crear una plataforma móvil para la enseñanza virtual. Para ello se intentarán conseguir, por orden, los siguientes objetivos:

- Conseguir un almacenamiento extra en el router para poder soportar todas las aplicaciones necesarias para llevar a cabo el proyecto.

- Instalar un sistema operativo de código abierto para poder personalizar el router y ajustarlo a las necesidades del proyecto.

- Instalar un servidor web ligero y eficiente que sea capaz de servir páginas html.

- Conseguir que el servidor de soporte a PHP para poder manejar páginas web de contenido dinámico.

- Instalar una base de datos para dar soporte a los cursos del “aula virtual”

- Crear un aula virtual, un espacio en la web donde los docentes pueden subir su material. Se plantean dos opciones, una es moodle, la conocida plataforma utilizada por la upct y wordpress, un sistema de gestión de contenido dedicado a la creación de blogs.

3 CONFIGURACIÓN HARDWARE

3.0 Escenario

Para poder instalar toda la parte software en el router, se debe primero ampliar la capacidad de almacenamiento del mismo (por defecto lleva 16 MB), ya que se necesita instalar un servidor http, una base de datos y sistema de gestión de contenidos para crear blogs llamado wordpress, todo ello más todos los paquetes y extensiones que necesitan para funcionar, ocupa mucho más de 16MB, por lo que sería imposible llevarlo a cabo sin una unidad de almacenamiento externa. Para ello, éste proyecto aborda el problema realizando un *hacking* del router, acoplando un slot de lectura de tarjetas SD a la placa base, de tal manera que el router pasará a tener una memoria del tamaño de la tarjeta SD (para el caso de éste PFC la tarjeta es de 2GB), con lo que ya se puede trabajar de manera holgada si hablamos de almacenamiento.

3.1 Material Utilizado

- Router *Linksys* WRT54GL
- Lector de tarjetas usb sd/sdhc/mmc
- Tarjeta SD *Kingston* 2GB
- Soldador de estaño
- Cable UTP CAT-5

3.1.1 Especificaciones Técnicas Router WRT54GLv1.1

El modelo que hemos utilizado ha sido el WRT54GLv1.1, en la sección 3.3.2 viene detallado como saber la versión hardware del router que tenemos.

Tipo de dispositivo	Enrutador inalámbrico
Factor de forma	Externo
Anchura	18.6 cm
Profundidad	20 cm
Altura	4.8 cm
Peso	482 g
Conmutador integrado	Conmutador de 4 puertos
Velocidad de transferencia de datos	54 Mbps
Banda de frecuencia	2.4 GHz
Protocolo de interconexión de datos	Ethernet, Fast Ethernet, IEEE 802.11b, IEEE 802.11g
Protocolo de conmutación	Ethernet
Nº de canales	13

seleccionables	
Indicadores de estado	Estado puerto, actividad de enlace, alimentación
Cantidad de antenas	2
Directividad	Omnidireccional
Iterfaces	1 x red - Ethernet 10Base-T/100Base-TX - RJ-45 (WAN) ¦ 4 x red - Ethernet 10Base-T/100Base-TX - RJ-45 ¦ 1 x red - Radio-Ethernet
Cables (Detalles)	1 x cable de red
Algoritmo de cifrado	AES, TKIP, WPA, WPA2
Cumplimiento de normas	CE, IC CS-03, FCC
Dispositivo de alimentación	Adaptador de corriente - externa
Temperatura mínima de funcionamiento	0 °C
Temperatura máxima de funcionamiento	40 °C
Ámbito de humedad de funcionamiento	10 - 85%

3.1.2 Especificaciones Técnicas Tarjeta Micro SD 2GB Kingston

El modelo utilizado es una micro sd de 2GB marca *Kingston* con adaptador SD.

Especificaciones técnicas:

* Capacidades: 2GB
* Dimensiones: 11 mm x 15 mm x 1 mm
* Peso: 2gr
* Homologada: Conforme con las especificaciones para las tarjetas 1.10 de la SD Card Association
* Compatible: con dispositivos SD en combinación con el adaptador
* Versáti: Puede funcionar como una tarjeta SD estándar al combinarla con el adaptador
* Ultra-Portátil: diseño extremadamente pequeño
* Sencillo: Tan fácil como conectar y listo (plug & play)
* Consumo: Bajo consumo energético para prolongar la vida de la batería de sus dispositivos
* Temperatura de funcionamiento: Entre -25° y 80° C / -13° y 185° F
* Temperatura de almacenamiento: Entre -40° y 85° C / -40° y 185° F

3.2 Instalación del Slot para Lectura de Tarjetas SD

3.2.1 Preparando y Conociendo el Escenario

Para que nuestro router sea capaz de leer tarjetas SD deberemos soldar el slot de lectura SD a la placa base de nuestro router, para ello debemos conocer qué hace cada pin de la tarjeta SD para saber a donde conectarlo exactamente en la placa base de nuestro linksys.
El primer paso será desmontar el lector de tarjetas y quedarnos únicamente con la parte que nos sirve (exclusivamente el slot donde introducimos la tarjeta), desoldando componentes como el USB y demás componentes que no nos serán necesarios para éste proyecto y facilitar así la instalación dentro de nuestro router. En la *Figura 1* podemos ver un slot lector de tarjetas desmontado a falta de desoldar el USB.

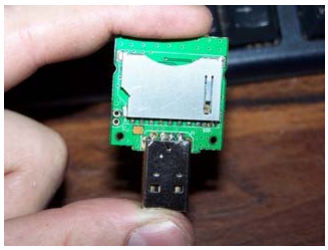

Figura 1. Slot lector de SD a falta de desoldar el USB.

Ésta será la pieza que finalmente integremos en nuestro router, cuya instalación viene detallada en el apartado 2.2.4.
Una tarjeta SD tiene 9 pines, en la *figura 2* podemos observarlos numerados.

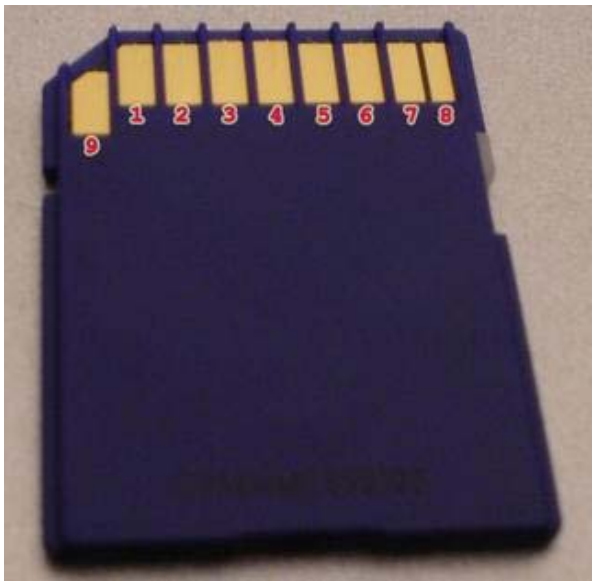

Figura 2. Pinouts de una tarjeta SD

La función de cada pinout es la siguiente:

Pin 1 – CS, Chip Select
Pin 2 – DI, Data In
Pin 3 – VSS, Ground
Pin 4 – VDD, 3.3v
Pin 5 – CLK, Clock
Pin 6 – VSS2, Ground
Pin 7 – DO, Data Out

Es interesante saber esto antes de pasar a las soldaduras, pues cada pin hay que soldarlo a un GPIO determinado en nuestra placa.

Los GPIO que utilizaremos serán los que aparecen destacados en las *figuras 3, 4 y 5.*

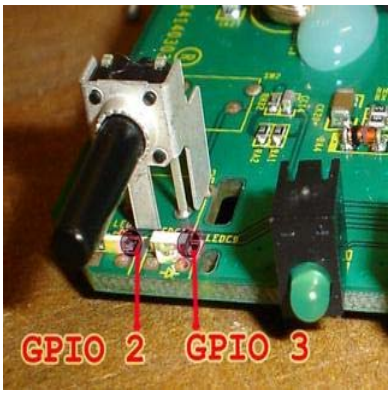

Figura 3. Gpio 2, gpio 3 en la parte frontal izquierda de la placa vista desde arriba.

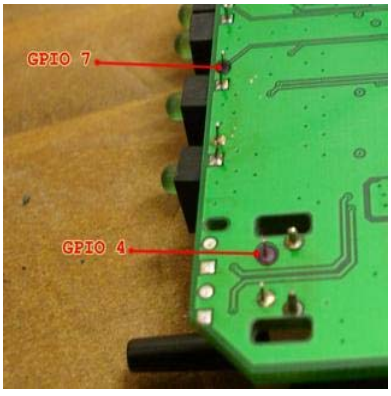

Figura 4. Gpio 4 y gpio 7 en la parte superior izquierda de la placa vista desde abajo.

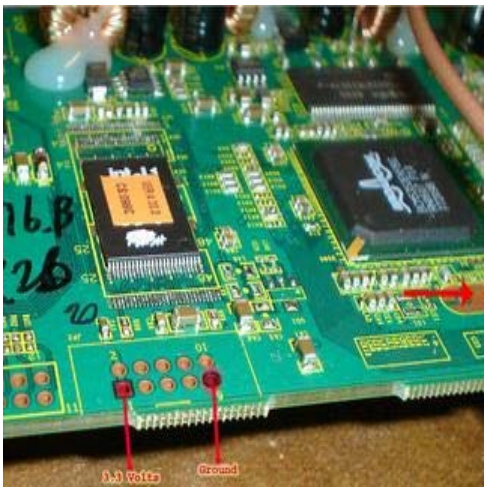

Figura 5. Alimentación y masa en la parte superior de la placa.

3.2.2 Soldaduras

Una vez conocidos los GPIO involucrados y los pinouts de la SD, procederemos a la conexión de los pinouts con los GPIO de la placa utilizando la *figura 4* como referencia.

Description	GPIO
Pin 1 - CS, Chip select	GPIO 7
Pin 2 - DI, Data in	GPIO2
Pin 3 - VSS, Ground	GND
Pin 4 - VDD, 3.3v	3.3v
Pin 5 - CLK, Clock	GPIO3
Pin 6 - VSS2, Ground	GND
Pin 7 - DO, Data out	GPIO4

Figura 6. Correspondencia Pinout – Gpio.

La manera más sencilla es abriendo un cable UTP CAT-5 y sacando cables de colores para facilitar la identificación, calculando la medida de los cables hasta el slot lector de tarjetas, el cual irá situado sobre el zócalo de conexiones de red del router.

3.2.3 Comprobando Conexiones

Éste paso es bastante importante, antes de seguir adelante, se deberá comprobar con un voltímetro en continua si las soldaduras se han realizado correctamente, por lo que mediremos desde el pinout hasta su correspondiente GPIO y nos aseguraremos que circula corriente correctamente de un punto a otro.

3.2.4 Integrando el Lector en el Router

Una vez realizadas las conexiones y comprobado que las soldaduras están correctas, se procede a pegar con silicona el lector de tarjetas sd al zócalo de conexiones de red del router.
Con una *dremel* se practica un corte en la carcasa del router a la altura del slot con 1mm de más a cada lado y 0,5mm arriba y abajo para que la tarjeta entre con holgura.

4 CONFIGURACIÓN SOFTWARE

4.1 Instalación de Openwrt

4.1.1 ¿Qué es Openwrt?

OpenWrt fue lanzado en Enero de 2004 por un equipo llamado *OpenWrt Project team*. Antes de OpenWrt, muchos utilizaban el firmware de serie basado en Linux de Linksys y lo personalizaban para un objetivo en especial. Esto fue lo que impulsó al equipo a desarrollar openwrt, un diseño mucho mejor e infinitamente más personalizable que el firmware de serie.
Diseñaron el núcleo basado en Linux/GNU con las mínimas prestaciones, básicamente adaptando Linux al procesador del WRT54G y a sus interfaces de red, también crearon un repositorio de paquetes debian, el cual se sigue actualizando, con los cuales el usuario final puede personalizar cada instalación para que encaje con sus necesidades.
El resultado es un firmware embedido extremadamente ligero, rápido y altamente personalizable.
Consiguieron desarrollar todo ello con el *OpenWrt software development kit (SDK).*

4.1.2 Eligiendo la Versión Correcta

La versión firmware a instalar depende en gran medida del hardware que tenemos, es muy importante aplicar la versión correcta, pues, en caso contrario, nos puede quedar un router completamente inutilizado.

Para saber la versión, sencillamente vamos a mirar en la parte trasera de nuestro linksys y observaremos un número de serie, según los cuatro primeros caracteres podremos conocer la versión hardware que tenemos.

En la figura 7 aparecen todas las versiones posibles según el número de serie:

Serial Number Prefix	Router/Version
WRT54G	
CDFF	WRT54G v.8.0
CDF0, CDF1	WRT54G v1.0
CDF2, CDF3	WRT54G v1.1
CDF5	WRT54G v2.0
WRT54G	
CDF7	WRT54G v2.2
CDF8	WRT54G v3.0
CDF9	WRT54G v3.1
CDFA	WRT54G v4.0
CDFB	WRT54G v5.0
CDFC	WRT54G v5.1
CDFD	WRT54G v6.0
CDFE	WRT54G v7.0
WRT54GL	
CL7A	WRT54GL v1.0
CL7B	WRT54GL v1.1
WRT54GS	
CGN0, CGN1	WRT54GS v1.0
CGN2	WRT54GS v1.1
CGN3	WRT54GS v2.0
CGN4	WRT54GS v2.1
CGN5	WRT54GS v3.0
CGN6	WRT54GS v4.0
CGN7	WRT54GS v5.0
CGN8	WRT54GS v5.1
CGN9	WRT54GS v6.0
WRTSL54GS	
CJK0	WRTSL54GS v1.0

Figura 7. Versiones hardware en función del número de serie.

En nuestro caso nuestro router lleva una versión hardware 1.1

4.1.3 Descargar el Firmware Válido para Nuestra Versión

Una vez que conocemos nuestra versión hardware, podremos elegir la versión del firmware a instalar sin miedo a dejar el router inutilizado por una firmware incorrecto.

Vamos al repositorio oficial de openwrt http://downloads.openwrt.org/ , de las versiones firmware hemos elegido kamikaze, y más concretamente la 8.09.2 pues la mayoría de referencias hablan de ésta versión como la más adecuada ya que tiene un repositorio mayor con más packages útiles para soportar un servidor http y una base de datos.

http://downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4/openwrt-wrt54g-squashfs.bin

4.2 Instalación de Kamikaze 8.09.2

4.2.1 Instalación vía Web

Éste es el método de instalación más sencillo y ha sido utilizado en éste proyecto.
Para la instalación vía web se deben seguir los siguientes pasos:

- Descargar la imagen firmware (apartado 3.1.3)
- Abrir un navegador web e ir a la dirección ip que tenga el router (por defecto 192.168.1.1).
- En la interfaz web del router click a Administration -> Firmware upgrade
- En upload image, se elige la imagen a instalar y se hace click al botón upgrade.
- Tras un par de minutos el router se reinicia sólo al finalizar la instalación.
- El router debería estar accesible por telnet en la dirección 192.168.1.1. También está disponible la nueva interfaz web (LUCI) en el navegador en http://192.168.1.1 . **Telnet quedará desactivado y ssh activado una vez que se haya configurado un password**.
- Una vez que haya sido habilitado ssh, es importante teclear los comandos que aparecen en la *figura 8* por si el router queda inutilizado y se pueda acceder a él a través de TFTP, en versiones hardware como la que nos concierne hay que utilizar *"wait_time"* en lugar de *"boot_time"*

```
nvram set boot_wait=on
nvram set boot_time=10
nvram commit && reboot
```

Figura 8. Comandos para habilitar tftp en el arranque.

Por comodidad, para éste proyecto le hemos asignado al router la dirección IP 192.168.1.2 pues el router de la estación de trabajo tiene la 192.168.1.1 y poder conectarnos así por wifi, pues toda la instalación se ha realizado con un cable de red directamente conectado a nuestro ordenador.
Esto lo hacemos conectando por el navegador a la dirección 192.168.1.1 y nos aparecerá la interfaz LUCI para configurar el router, en modo administrador podremos cambiar la dirección ip de nuestro router a la 192.168.1.2 y poder trabajar con conexión a internet y utilizando el linksys como un dispositivo de red local.

Una vez reiniciado el router, el router queda accesible desde el terminal escribiendo:

ssh root@192.168.1.2

Aparecerá una pantalla de bienvenida como la que aparece en la *figura 9*

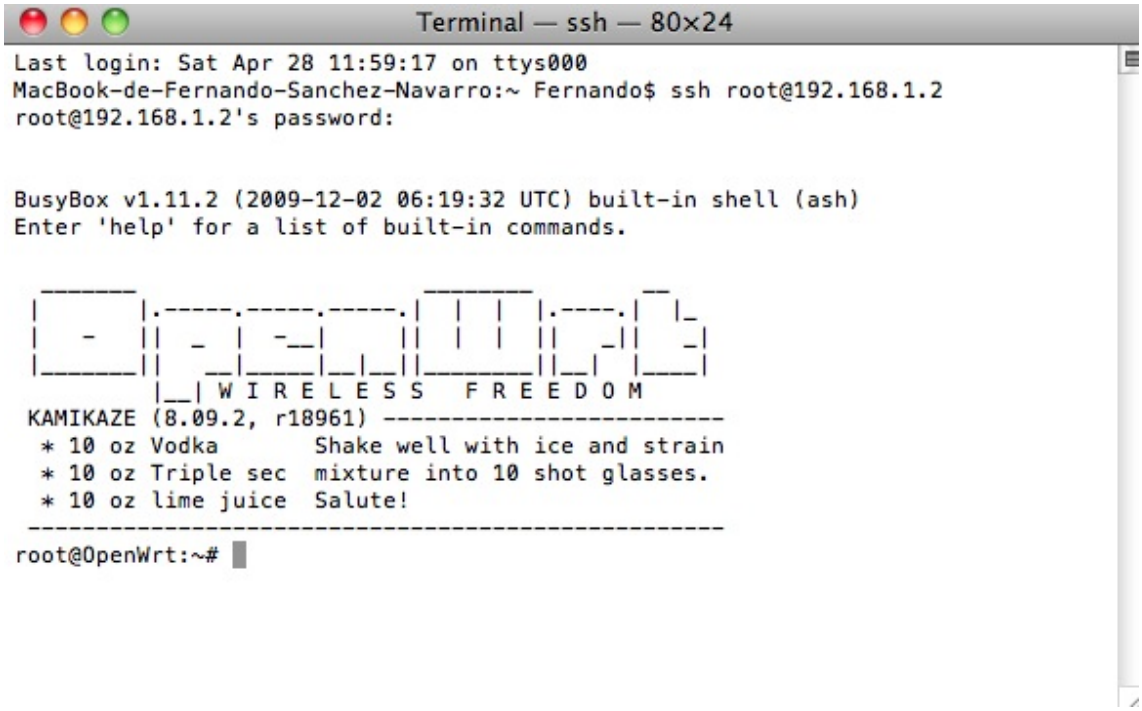

Figura 9. Pantalla de bienvenida de kamikaze 8.09.2

Esto quiere decir que la instalación se ha realizado correctamente y el router ya está preparado para instalar paquetes personalizados del repositorio oficial de openwrt en la dirección http://downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4/packages/ (broadcom y versión kernel 2.4)

4.2.2 Instalación vía TFTP

Primero se debe configurar TFTP para que sea inicializado en el arranque (*ver figura 11)*
Guardamos la imagen en el directorio root y se escriben los siguientes comandos:

```
tftp 192.168.1.1
binary
trace
rexmt 1
put openwrt-wrt54g-squashfs.bin
```

Al reiniciar el router queda accesible vía web y ssh.

4.3 Configuración del Driver Lector de Tarjetas

4.3.1 Driver

El potencial del router wrt54gl con un sistema operativo Linux de código abierto como openwrt es muy alto, hay numerosos paquetes creados por la comunidad de desarrolladores de openwrt para hacer del router algo más que un simple router.
Uno de los paquetes más interesantes que se ha desarrollado se llama *kmod-broadcom-mmc,* para la versión 7.07 de kamikaze en adelante. Un paquete que nos permite leer de una tarjeta externa, como es nuestro caso y poder así ampliar las posibilidades.
Éste paquete instala numerosos archivos que nos permiten leer de la tarjeta, pero es demasiado genérico y en numerosos casos no funciona. Podemos encontrar una versión optimizada y relativamente actualizada en la siguiente dirección:

http://wiki.openwrt.org/_media/oldwiki/openwrtdocs/customizing/hardware/mmc-v1.3.4-gpio2.tgz

Éste paquete contiene únicamente un archivo llamado mmc.o que es el driver que nos permite leer la tarjeta.
Hay otra versión del driver llamada mmc-v1.3.4-gpio5.tgz, pero es para una versión distinta de placa base, nosotros utilizamos el GPIO2, pues el driver para el gpio2 se utiliza en routers WRT54G/WRT54GL y el gpio5 sólo hasta la versión 3.1 del WRT54G, no GL.

4.3.2 Instalación de los Módulos Necesarios

A continuación se detallan los módulos necesarios para que seamos capaces de leer de la tarjeta. Todos los podemos encontrar dentro del repositorio de nuestra versión de kamikaze (http://downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4/packages/)

- kmod-broadcom-mmc (Driver lector mmc)
- kmod-fs-ext2 (para leer tarjetas con formato ext2)
- kmod-fs-ext3 (para leer tarjetas con formato ext3)
- libext2fs (librerías ext2)
- kmod-fs-vfat (para tarjetas VFAT)

Conectamos con el router con ssh, siempre siendo superusuario:

ssh root@192.168.1.2

Una vez dentro, procedemos a la instalación en primer lugar del driver con la siguiente instrucción:

Opkg install kmod-broadcom-mmc
Opkg es la equivalencia en openwrt al comando ipkg de Linux.
Con ésta instrucción el router va directamente a la carpeta packages del repositorio de nuestra versión kamikaze, pues va incluida la dirección en el archivo opkg.conf situado

en la carpeta /etc/opkg.conf desde el raíz de nuestro router, más concretamente en la primera línea de éste archivo con la instrucción "src/gz snapshots http://downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4/packages"

Si quisiéramos instalar packages de otras versiones (alguna vez pueden ser compatibles) deberíamos cambiar ésta dirección en el archivo de configuración del opkg.

Los packages también se pueden instalar indicando directamente la URL donde está el package, así:

Opkg install http://downloads.openwrt.org/kamikaze/8.09.2/brcm-2.4/packages/kmod-broadcom-mmc_2.4.35.4+0.1-brcm-2.4-1_mipsel.ipk

Instalamos el resto de packages para leer varios formatos de tarjeta:

Opkg install kmod-fs-ext2
Opkg install kmod-fs-ext3
Opkg install libext2fs
Opkg install kmod-fs-vfat

Para los que sea necesario instalará packages dependientes.

4.3.3 Configurando los Módulos Instalados

Una vez instalados los módulos hay que configurarlos y adaptarlos a nuestro caso particular.
Lo primero es substituir el driver mmc.o por el optimizado indicado en el apartado 3.3.1
Éste driver se encuentra en /lib/modules/2.4.35.4/mmc.o
Desde el terminal, podemos copiarlo por scp mediante la siguiente instrucción:

scp /ruta_al_archivo/mmc.o root@192.168.1.2:/lib/modules/2.4.35.4/mmc.o

De ésta manera el obsoleto driver de nuestra versión de kamikaze será reemplazado por el optimizado creado por la comunidad de desarrolladores de openwrt.

Se define el punto de montaje en el archivo **etc/config/fstab** con el siguiente formato:

config mount
option target /mnt/sd
option device /dev/mmc/disc0/part1
option fstype ext2
option options rw,sync,noatime
option enabled 1
Con ésta instrucción en el archivo *fstab* el sistema montará automáticamente la unidad en /mnt/sd con formato ext2.

Es muy importante colocar la opción **echo "0x9c" > /proc/diag/gpiomask** la explicación se debe a lo siguiente:
Al realizar la modificación de la tarjeta SD, se están utilizando las líneas GPIO disponibles en el router. OpenWRT utiliza estas líneas para entrar en un estado denominado “failsafe mode”. Por lo tanto, se debe aplicar una máscara, que debe ser introducida en el archivo /proc/diag/gpiomask. Esta máscara define que líneas GPIO se están utilizando, y que no deben ser tenidas en cuenta para entrar al modo anteriormente mencionado. Se utilizó el hexadecimal 0x9C, debido a que se están utilizando las líneas GPIO 2, 3, 4, 7. Asumiendo un byte, se coloca un 1 a cada posición de línea GPIO que se vaya a utilizar, por lo tanto resulta 10011100, que en su equivalente hexadecimal es justamente 0x9C. Se menciona esto puesto que algunas modificaciones para utilizar la tarjeta SD, usan otras líneas GPIO, por lo tanto es común ver otras máscaras utilizadas.

Creamos un script de arranque en etc/config llamado bootfromsd con las directivas que queremos que el sistema ejecute al iniciar:

option target '/mnt/sd'
option device '/dev/mmc/disco0/part1'
option gpiomask '0x9c'
option modules 'mmc ext2' (nos cargará los módulos mmc y ext2 al iniciar)
option enabled '1'

Al reiniciar si ejectuamos *lsmod* nos aparece una lista con los módulos cargados.
Ejecutamos y vemos en la *figura 10* como, efectivamente, el sistema nos ha cargado los módulos mmc y ext2 que nos permitirá leer tarjetas con formato ext2.

```
root@OpenWrt:/etc/config# lsmod
Module                  Size  Used by    Tainted: P
wlcompat                9504   0 (unused)
ip_conntrack_tftp       1628   0 (unused)
ip_nat_irc              2296   0 (unused)
ip_conntrack_irc        3028   1
ip_nat_ftp              2920   0 (unused)
ip_conntrack_ftp        4172   1
ipt_MASQUERADE          1316   1
iptable_nat            20856   3 [ip_nat_irc ip_nat_ftp ipt_MASQUERADE]
ipt_state                408   6
ip_conntrack           22368   3 [ip_conntrack_tftp ip_nat_irc ip_conntrack_irc ip_nat_ftp ip_conntrack_ftp
ipt_MASQUERADE iptable_nat ipt_state]
ipt_REJECT              3932   2
ipt_TCPMSS              2316   1
ipt_LOG                 3804   0 (unused)
ipt_multiport            748   0 (unused)
ipt_mac                  556   0 (unused)
ipt_limit                892   1
iptable_mangle          2156   0 (unused)
iptable_filter          1676   1
ip_tables              16960  13 [ipt_MASQUERADE iptable_nat ipt_state ipt_REJECT ipt_TCPMSS ipt_LOG ipt_mul
tiport ipt_mac ipt_limit iptable_mangle iptable_filter]
ext2                   40304   1
ppp_async               8044   0 (unused)
ppp_generic            22380   0 [ppp_async]
slhc                    6064   0 [ppp_generic]
mmc                    26540   1
wl                    666560   0 (unused)
switch-robo             5180   0 (unused)
switch-core             5104   0 [switch-robo]
diag                   50448   0 (unused)
root@OpenWrt:/etc/config#
```

Figura 10. Ejecución del comando lsmod en la línea de comandos.

Ejecutamos *dmesg* y nos aparece como ha cargado la tarjeta correctamente:

```
root@OpenWrt:~# dmesg
CPU revision is: 00029008
Primary instruction cache 16kB, physically tagged, 2-way, linesize 16 bytes.
Primary data cache 8kB, 2-way, linesize 16 bytes.
Linux version 2.4.35.4 (agb@arrakis) (gcc version 3.4.6 (OpenWrt-2.0)) #12 Tue Dec
29 15:30:20 UTC 2009
Setting the PFC to its default value
Determined physical RAM map:
 memory: 01000000 @ 00000000 (usable)
On node 0 totalpages: 4096
zone(0): 4096 pages.
zone(1): 0 pages.
zone(2): 0 pages.
Kernel command line: root=/dev/mtdblock2 rootfstype=squashfs,jffs2 init=/etc/preinit
noinitrd console=ttyS0,115200
CPU: BCM5352 rev 0 at 200 MHz
Using 100.000 MHz high precision timer.
Calibrating delay loop... 199.47 BogoMIPS
Memory: 14244k/16384k available (1437k kernel code, 2140k reserved, 100k data, 84k
init, 0k highmem)
Dentry cache hash table entries: 2048 (order: 2, 16384 bytes)
Inode cache hash table entries: 1024 (order: 1, 8192 bytes)
Mount cache hash table entries: 512 (order: 0, 4096 bytes)
Buffer cache hash table entries: 1024 (order: 0, 4096 bytes)
Page-cache hash table entries: 4096 (order: 2, 16384 bytes)
Checking for 'wait' instruction...  unavailable.
POSIX conformance testing by UNIFIX
PCI: no core
PCI: Fixing up bus 0
Linux NET4.0 for Linux 2.4
Based upon Swansea University Computer Society NET3.039
Initializing RT netlink socket
Starting kswapd
Registering mini_fo version $Id$
devfs: v1.12c (20020818) Richard Gooch (rgooch@atnf.csiro.au)
devfs: boot_options: 0x1
JFFS2 version 2.1. (C) 2001 Red Hat, Inc., designed by Axis Communications AB.
squashfs: version 3.0 (2006/03/15) Phillip Lougher
pty: 256 Unix98 ptys configured
Serial driver version 5.05c (2001-07-08) with MANY_PORTS SHARE_IRQ
SERIAL_PCI enabled
ttyS00 at 0xb8000300 (irq = 3) is a 16550A
ttyS01 at 0xb8000400 (irq = 3) is a 16550A
b44.c:v0.93 (Mar, 2004)
PCI: Setting latency timer of device 00:01.0 to 64
eth0: Broadcom 47xx 10/100BaseT Ethernet 00:23:69:2a:0f:a3
Physically mapped flash: Found an alias at 0x400000 for the chip at 0x0
Physically mapped flash: Found an alias at 0x800000 for the chip at 0x0
Physically mapped flash: Found an alias at 0xc00000 for the chip at 0x0
Physically mapped flash: Found an alias at 0x1000000 for the chip at 0x0
```

Physically mapped flash: Found an alias at 0x1400000 for the chip at 0x0
Physically mapped flash: Found an alias at 0x1800000 for the chip at 0x0
Physically mapped flash: Found an alias at 0x1c00000 for the chip at 0x0
 Amd/Fujitsu Extended Query Table v1.1 at 0x0040
number of CFI chips: 1
cfi_cmdset_0002: Disabling fast programming due to code brokenness.
Flash device: 0x400000 at 0x1c000000
bootloader size: 262144
Physically mapped flash: Filesystem type: squashfs, size=0x1820c3
Creating 5 MTD partitions on "Physically mapped flash":
0x00000000-0x00040000 : "cfe"
0x00040000-0x003f0000 : "linux"
0x000bc000-0x00240000 : "rootfs"
mtd: partition "rootfs" doesn't start on an erase block boundary -- force read-only
0x003f0000-0x00400000 : "nvram"
0x00240000-0x003f0000 : "rootfs_data"
Initializing Cryptographic API
NET4: Linux TCP/IP 1.0 for NET4.0
IP Protocols: ICMP, UDP, TCP, IGMP
IP: routing cache hash table of 512 buckets, 4Kbytes
TCP: Hash tables configured (established 1024 bind 2048)
Linux IP multicast router 0.06 plus PIM-SM
NET4: Unix domain sockets 1.0/SMP for Linux NET4.0.
NET4: Ethernet Bridge 008 for NET4.0
802.1Q VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
All bugs added by David S. Miller <davem@redhat.com>
VFS: Mounted root (squashfs filesystem) readonly.
Mounted devfs on /dev
Freeing unused kernel memory: 84k freed
Algorithmics/MIPS FPU Emulator v1.5
diag: Detected 'Linksys WRT54G/GS/GL'
b44: eth0: Link is up at 100 Mbps, full duplex.
b44: eth0: Flow control is off for TX and off for RX.
roboswitch: Probing device eth0: found!
mini_fo: using base directory: /
mini_fo: using storage directory: /jffs
jffs2.bbc: SIZE compression mode activated.
b44: eth0: Link is up at 100 Mbps, full duplex.
b44: eth0: Flow control is off for TX and off for RX.
eth0.0: add 01:00:5e:00:00:01 mcast address to master interface
eth0.0: dev_set_promiscuity(master, 1)
device eth0 entered promiscuous mode
device eth0.0 entered promiscuous mode
br-lan: port 1(eth0.0) entering learning state
br-lan: port 1(eth0.0) entering forwarding state
br-lan: topology change detected, propagating
eth0.1: add 01:00:5e:00:00:01 mcast address to master interface
PCI: Setting latency timer of device 00:05.0 to 64
PCI/DMA

wl0: wlc_attach: chiprev 0 coreunit 0 corerev 9 cccap 0x4787aa maccap 0x0 band 2.4G, phy_type 2 phy_rev 7 ana_rev 2
wl0: Broadcom BCM4320 802.11 Wireless Controller 4.150.10.5
[INFO] mmc_hardware_init: initializing GPIOs
[INFO] mmc_card_init: the period of a 380KHz frequency lasts 524 CPU cycles
[INFO] mmc_card_init: powering card on. sending 80 CLK
[INFO] mmc_card_init: 80 CLK sent in 43646 CPU cycles
[INFO] mmc_card_init: resetting card (CMD0)
[INFO] mmc_card_init: doing initialization loop
[INFO] mmc_card_init: card inited successfully in 1501 tries (46305720 CPU cycles).
[INFO] mmc_init: MMC/SD Card ID:
02 54 4d 53 41 30 32 47 05 23 ab 1f c5 00 b3 79 [INFO] Manufacturer ID : 02
[INFO] OEM/Application ID: TM
[INFO] Product name : SA02G
[INFO] Product revision : 0.5
[INFO] Product SN : 23ab1fc5
[INFO] Product Date : 2011-3
[INFO] mmc_card_config: size = 1927168, hardsectsize = 1024, sectors = 1927168
[WARN] mmc_init: hd_sizes=1927168, hd[0].nr_sects=3854336
[INFO] mmc_card_init: set_blocklen (CMD16) succeeded !
Partition check:
mmca: p1
CSLIP: code copyright 1989 Regents of the University of California
PPP generic driver version 2.4.2

Al ejecutar el comando df nos aparecerá la tarjeta sd montada en el directorio /mnt /sd como podemos ver en la *figura 11*

```
BusyBox v1.11.2 (2009-12-02 06:19:32 UTC) built-in shell (ash)
Enter 'help' for a list of built-in commands.

  _______                     ________        __
 |       |.-----.-----.-----.|  |  |  |.----.|  |_
 |   -   ||  _  |  -__|     ||  |  |  ||   _||   _|
 |_______||   __|_____|__|__||________||__|  |____|
          |__| W I R E L E S S   F R E E D O M
 KAMIKAZE (8.09.2, r18961) ------------------------
  * 10 oz Vodka       Shake well with ice and strain
  * 10 oz Triple sec  mixture into 10 shot glasses.
  * 10 oz lime juice  Salute!
 ---------------------------------------------------
root@OpenWrt:~# df
Filesystem           1k-blocks      Used Available Use% Mounted on
rootfs                    1600      1600         0 100% /
/dev/root                 1600      1600         0 100% /rom
tmpfs                     7164        52      7112   1% /tmp
/dev/mtdblock/4           1728       496      1232  29% /jffs
mini_fo:/jffs             1600      1600         0 100% /
/dev/mmc/disc0/part1   1895440   1233596    565560  69% /mnt/sd
root@OpenWrt:~#
```

Figura 11. Captura de pantalla al ejecutar el comando df.

Una vez montada la tarjeta y configurado el sistema para que la monte al iniciar, ya tenemos el espacio necesario para poder instalar una gran cantidad de paquetes.

4.3.4 Cambiar la Ruta de Instalación por Defecto

Cuando se utiliza el comando *opkg*, los paquetes son instalados por defecto en el directorio raíz del router, por lo que si no nos damos cuenta, se puede sobrepasar la memoria disponible y dejaríamos el router inutilizado y tendríamos que volver a empezar. Esto es un bug de openwrt, cuando la memoria se llena, aunque se libere espacio, el sistema queda bloqueado y no nos permite ejecutar ningún comando.

Para ello debemos editar el archivo de configuración del *opkg*. Éste archivo se encuentra situado en /etc/opkg.conf. Simplemente se edita y se añande la línea:

dest sd /mnt/sd

Cuando queremos instalar un paquete utilizando *opkg*, hay una opción para indicar el destino, se realiza de la siguiente manera:

opkg –d ***identificador*** *install paquete_a_instalar*

En éste caso, al ejecutar ésta línea, opkg va a buscar a su archivo de configuración el identificador sd y encuentra de ésta forma su correspondiente ruta de instalación, en éste caso /mnt/sd.

4.4 Instalación del Servidor Web (Lighttpd)

4.4.1 Introducción

Para poder instalar wordpress, necesitaremos un servidor, deberemos configurar php para poder interpretar las páginas y una base de datos que gestione nuestro blog. Cabe decir que lo que propone éste PFC es la integración de todo el sistema a un bajo costo, por lo que el servidor web será integrado directamente en nuestra plataforma, por supuesto, el potencial hardware del router es algo limitado para soportar todo lo mencionado, pero, tener contratado un servidor web externo supondría un costo extra para nada contemplado por ésta propuesta. La plataforma será totalmente funcional, pero algo lenta dados los medios técnicos y económicos.
Existe un paquete llamado *lighttpd*, es un servidor ligero, preparado para sistemas con muy pocos recursos que funciona rápido y sencillo.

4.4.2 Instalación

La instalación de lighttpd en el router wrt54gl, es sencilla, utilizaremos la siguiente instrucción:

opkg –d sd install lighttpd lighttpd-mod-cgi lighttpd-mod-status

De ésta manera se instalará primero el paquete general de lighttpd (versión 1.4) en el directorio /mnt/sd, y siguiendo el orden se instalarán el resto de los paquetes. Obviamente, instalará los paquetes dependientes si el instalador lo cree necesario.
Una vez instalados es importante comprobar que se han instalado correctamente los módulos en la carpeta /mnt/sd/usr/lib/lighttpd y habilitarlos en el archivo de configuración lighttpd.conf
Navegamos a la carpeta indicada y comprobamos que, efectivamente, se han instalado los módulos en la carpeta (ver *figura 12*).

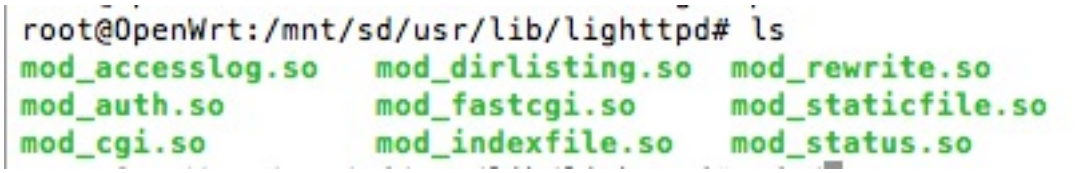

Figura 12. Captura de pantalla al ejecutar el comando df.

Vamos al archivo de configuración del lighttpd situado en /mnt/sd/etc/lighttpd.conf y quitamos la # a los módulos que hemos instalado y queremos arrancar, justo al principio del archivo de configuración:

```
# lighttpd configuration file
#
## modules to load
# all other module should only be loaded if really neccesary
# - saves some time
# - saves memory
server.modules = (
#       "mod_rewrite",
#       "mod_redirect",
#       "mod_alias",
        "mod_auth",
        "mod_status",
#       "mod_setenv",
#       "mod_fastcgi",
#       "mod_proxy",
#       "mod_simple_vhost",
        "mod_cgi",
#       "mod_ssi",
#       "mod_usertrack",
#       "mod_expire"
)

# force use of the "write" backend (closes: #2401)
- lighttpd.conf 11/225 4%
```

De ésta manera cuando el servidor arranque, los módulos se cargarán en memoria y estarán funcionando.

4.4.3 Configuración

Hay que modificar el archivo de configuración lighttpd.conf para ajustarlo a nuestras necesidades, marcadas en negrita, las modificaciones realizadas por el autor del PFC.

```
# lighttpd configuration file
#
## modules to load
# all other module should only be loaded if really neccesary
# - saves some time
# - saves memory
server.modules = (
#     "mod_rewrite",
#     "mod_redirect",
#     "mod_alias",
      "mod_auth",
      "mod_status",
#     "mod_setenv",
#     "mod_fastcgi",
#     "mod_proxy",
#     "mod_simple_vhost",
      "mod_cgi",
#     "mod_ssi",
#     "mod_usertrack",
#     "mod_expire"
)

# force use of the "write" backend (closes: #2401)
```

```
server.network-backend = "write"

server.event-handler = "poll"
## a static document-root, for virtual-hosting take look at the
## server.virtual-* options
server.document-root = "/mnt/sd/www"

## where to send error-messages to
server.errorlog = "/mnt/sd/var/log/lighttpd/error.log"
## files to check for if .../ is requested
index-file.names   =   ("index.php",   "index.html",   "default.html",
"index.htm", "default.htm" )

## mimetype mapping
mimetype.assign = (
      ".pdf"   => "application/pdf",
      ".class" => "application/octet-stream",
      ".pac"   => "application/x-ns-proxy-autoconfig",
      ".swf"   => "application/x-shockwave-flash",
      ".wav"   => "audio/x-wav",
      ".gif"   => "image/gif",
      ".jpg"   => "image/jpeg",
      ".jpeg"  => "image/jpeg",
      ".png"   => "image/png",
      ".css"   => "text/css",
      ".html"  => "text/html",
      ".htm"   => "text/html",
      ".js"    => "text/javascript",
      ".txt"   => "text/plain",
      ".dtd"   => "text/xml",
      ".xml"   => "text/xml"
 )

## Use the "Content-Type" extended attribute to obtain mime type if
possible
#mimetypes.use-xattr = "enable"

## send a different Server: header
## be nice and keep it at lighttpd
#server.tag = "lighttpd"

$HTTP["url"] =~ "\.pdf$" {
      server.range-requests = "disable"
}

##
# which extensions should not be handle via static-file transfer
#
# .php, .pl, .fcgi are most often handled by mod_fastcgi or mod_cgi
static-file.exclude-extensions = ( ".php", ".pl", ".fcgi" )

######### Options that are good to be but not neccesary to be changed
#######

## bind to port (default: 80)
server.port = 80

## bind to localhost (default: all interfaces)
#server.bind = "localhost"

## error-handler for status 404
```

```
#server.error-handler-404 = "/error-handler.html"
#server.error-handler-404 = "/error-handler.php"

## to help the rc.scripts
server.pid-file = "/mnt/sd/var/run/lighttpd.pid"


###### virtual hosts
##
##     If you want name-based virtual hosting add the next three
settings and load
##   mod_simple_vhost
##
## document-root =
##     virtual-server-root + virtual-server-default-host + virtual-
server-docroot or
##   virtual-server-root + http-host + virtual-server-docroot
##
#simple-vhost.server-root = "/mnt/sd/www/moodle/moodle/"
#simple-vhost.default-host = "192.168.1.2"
#simple-vhost.document-root = "/mnt/sd/www/moodledata/"


##
## Format: <errorfile-prefix><status>.html
## -> ..../status-404.html for 'File not found'
#server.errorfile-prefix = "/www/error-"

## virtual directory listings
#server.dir-listing = "enable"

## send unhandled HTTP-header headers to error-log
#debug.dump-unknown-headers = "enable"

### only root can use these options
#
# chroot() to directory (default: no chroot() )
#server.chroot = "/"

## change uid to <uid> (default: don't care)
server.username = "_www"
#
server.upload-dirs = ( "/tmp" )

## change uid to <uid> (default: don't care)
#server.groupname = "nobody"

#### compress module
#compress.cache-dir          = "/dev/null/"
#compress.filetype           = ("text/plain", "text/html")

#### proxy module
## read proxy.txt for more info
#proxy.server = (
#     ".php" => (
#          "localhost" => (
#               "host" => "192.168.0.101",
#               "port" => 80
#          )
#     )
#)
```

```
#### fastcgi module
## read fastcgi.txt for more info
#fastcgi.server = (
#     ".php" => (
#           "localhost" => (
#                "socket" => "/tmp/php-fastcgi.socket",
#                "bin-path" => "/usr/local/bin/php"
#           )
#     )
#)

#### CGI module
cgi.assign = ( ".php"  => "/mnt/sd/usr/bin/php-cgi")
#cgi.assign = ( ".pl" => "/usr/bin/perl",
# ".cgi" => "/usr/bin/perl" )

#### SSL engine
#ssl.engine = "enable"
#ssl.pemfile = "server.pem"

#### status module
status.status-url = "/status"
status.config-url = "/config"

#### auth module
## read authentification.txt for more info
#auth.backend = "plain"
#auth.backend.plain.userfile = "lighttpd.user"
#auth.backend.plain.groupfile = "lighttpd.group"
#auth.require = (
#     "/server-status" => (
#           "method"  => "digest",
#           "realm"   => "download archiv",
#           "require" => "group=www|user=jan|host=192.168.2.10"
#     ),
#     "/server-info" => (
#           "method"  => "digest",
#           "realm"   => "download archiv",
#           "require" => "group=www|user=jan|host=192.168.2.10"
#     )
#)

#### url handling modules (rewrite, redirect, access)
#url.rewrite = ( "^/$" => "/server-status" )
#url.redirect = ( "^/wishlist/(.+)" => "http://www.123.org/$1" )

#### both rewrite/redirect support back reference to regex conditional
using %n
#$HTTP["host"] =~ "^www\.(.*)" {
#     url.redirect = ( "^/(.*)" => "http://%1/$1" )
#}

#### expire module
#expire.url = ( "/buggy/" => "access 2 hours", "/asdhas/" => "access
plus 1 seconds 2 minutes")

#### ssi
#ssi.extension = ( ".shtml" )

#### setenv
```

```
#setenv.add-request-header  = ( "TRAV_ENV" => "mysql://user@host/db" )
#setenv.add-response-header = ( "X-Secret-Message" => "42" )

#### variable usage:
## variable name without "." is auto prefixed by "var." and becomes
"var.bar"
#bar = 1
#var.mystring = "foo"

## integer add
#bar += 1
## string concat, with integer cast as string, result: "www.foo1.com"
#server.name = "www." + mystring + var.bar + ".com"
## array merge
#index-file.names = (foo + ".php") + index-file.names
#index-file.names += (foo + ".php")

#### include
#include /etc/lighttpd/lighttpd-inc.conf
## same as above if you run: "lighttpd -f /etc/lighttpd/lighttpd.conf"
#include "lighttpd-inc.conf"

#### include_shell
#include_shell "echo var.a=1"
## the above is same as:
#var.a=1
```

Explicación de las modificaciones:

```
       "mod_auth",
       "mod_status",
#      "mod_setenv",
#      "mod_fastcgi",
#      "mod_proxy",
#      "mod_simple_vhost",
       "mod_cgi",
```

Hemos habilitado los módulos para que carguen cuando ejecutemos el script quitando la # (de ésta forma se procesan)

```
server.document-root = "/mnt/sd/www"
```

El paquete lighttpd viene preconfigurado para una instalación normal en el raíz del router, como nosotros lo hemos instalado en /mnt/sd, hay que cambiar numerosas rutas de instalación en algunos archivos de los paquetes. El primer caso es el *server.document-root*, el directorio raíz del servidor, a partir del cual empezará a leer los archivos alojados en el, si queremos acceder a una carpeta llamada “fotos” en nuestro servidor, tendríamos que alojarla en /mnt/sd/www/fotos y quedaría accesible en el navegador mediante http://192.168.1.2/fotos. Hemos indicado la dirección del router que será él mismo quien sirva las páginas web.

```
server.errorlog = "/mnt/sd/var/log/lighttpd/error.log"
```

server.errorlog, indicamos la ruta donde el servidor creará el archivo error.log donde mostrará todos los mensajes de error. Por defecto la ruta era /var/log/lighttpd/error.log

```
server.port = 80
```

Éste es el puerto en el que funcionará el servidor (server.port), realmente éste valor no se ha modificado, pero cabe destacar su funcionalidad.
Se podría haber mantenido la dirección 192.168.1.1 para nuestro linksys, podemos modificar el puerto por el que se ejecuta cada uno de los 2 servidores que tenemos ahora en funcionamiento (la interfaz LUCI, que por defecto utiliza el 80, y nuestro lighttpd que por defecto también utiliza el mismo puerto), al haber cambiado la dirección ip del router, no es necesario especificar el puerto, ya que tendremos en 192.168.1.2:80 el servidor lighttpd y en 192.168.1.1:80 nuestra interfaz de configuración del router (LUCI).
Si se quisiera tener los dos servidores en la misma dirección IP, se deben especificar distintos puertos para cada uno, si cambiamos la variable server.port a, por ejemplo, 8080, para una misma dirección quedarán accesibles 2 servidores web por distintos puertos.
Escribiendo en el navegador 192.168.1.1 ó 192.168.1.1:80 accederíamos a la interfaz LUCI, mientras que si escribimos 192.168.1.1:8080, en la misma dirección, pero distinto puerto, accederíamos al servidor lighttpd.

```
server.pid-file = "/mnt/sd/var/run/lighttpd.pid"
```

En lighttpd.pid es donde aparece el número de proceso de lighttpd.

```
cgi.assign = ( ".php"  => "/mnt/sd/usr/bin/php-cgi")
```

En ésta línea indicamos dónde se encuentra el procesador de páginas php, el cual se detallará más adelante en el apartado 3.5 Instalación de php.

4.4.4 Arranque Manual

Para arrancar el servidor, deberemos indicar dónde se encuentra el archivo de configuración mediante la opción –f:

/mnt/sd/usr/sbin/lighttpd start -f /mnt/sd/etc/lighttpd.conf

De ésta manera, el servidor arrancará y empezará a escuchar en el puerto 80. (El indicado en el archivo de configuración lighttpd.conf).

Para probarlo crearemos un archivo html “Hola mundo”, el cual está explicado en el punto 4.4.6

4.4.5 Primer "Hola mundo" …….

Para comprobar que el servidor está arrancado correctamente, crearemos un archivo .html en la ruta raíz del servidor (recordamos que es /mnt/sd/www).
Crearemos con el editor *vi* un archivo llamado index.html (el servidor lee por defecto los index.html del directorio al que accedeomos, y escribimos el siguiente código:

```
<html>
<title>HOLA MUNDO !!!!</title>
<body>
HoLa MunDo !!!
</body>
</html>
```

Abriendo el navegador, con el servidor lighttpd arrancado, escribimos en la barra de dirección 192.168.1.2 y nuestro servidor nos cargará nuestro primer .html

4.5 Instalación de PHP 5.2.6

4.5.1. ¿Qué es PHP y que uso le da éste proyecto?

PHP es un lenguaje de programación interpretado (Lenguaje de alto rendimiento), diseñado originalmente para la creación de páginas web dinámicas. Se usa principalmente para la interpretación del lado del servidor.

PHP nos permitirá interpretar las páginas web con contenido dinámico y a la misma vez nos servirá de enlace para la conexión con la base de datos (en éste caso PostgreSQL, detallado en el puton 4.6)

Obviamente, resulta necesario configurar php para que sea capaz de interpretar las páginas con contenido dinámico para albergar un blog.

4.5.2 Instalación

Básicamente la instalación de los paquetes es muy sencilla, se reduce a instalar los paquetes del repositorio:

opkg –d sd install php5 php5-cgi php5-mod-gd

Éste comando instalará los paquetes en /mnt/sd

4.5.3 Configuración

Una vez instalados los paquetes, deberemos modificar el archivo de configuración de php, llamado php.ini y situado en la carpeta /mnt/sd/etc/php.ini, para que se ajuste a las necesidades de éste PFC.

A continuación, se detalla el código fuente del archivo php.ini y en negrita, las modificaciones realizadas por el autor junto con su explicación.

```
[PHP]

cgi.fix_pathinfo=1

zend.ze1_compatibility_mode = Off

short_open_tag = On
asp_tags = Off
precision    =  12
y2k_compliance = On
output_buffering = On

zlib.output_compression = Off
;zlib.output_handler =
implicit_flush = Off

unserialize_callback_func=
serialize_precision = 100

allow_call_time_pass_reference = On

safe_mode = Off

safe_mode_gid = Off

safe_mode_include_dir =

safe_mode_exec_dir =

safe_mode_allowed_env_vars = PHP_

safe_mode_protected_env_vars = LD_LIBRARY_PATH

;open_basedir =

disable_functions =

disable_classes =

; Colors for Syntax Highlighting mode.  Anything that's acceptable in
; <span style="color: ???????"> would work.
;highlight.string  = #DD0000
;highlight.comment = #FF9900
;highlight.keyword = #007700
;highlight.bg      = #FFFFFF
;highlight.default = #0000BB
;highlight.html    = #000000

expose_php = On
```

```
;;;;;;;;;;;;;;;;;;
; Resource Limits ;
;;;;;;;;;;;;;;;;;;

//max_execution_time = 0     ; Maximum execution time of each script,
in seconds
max_execution_time = 1000 ;

//Debido a los limitados recursos de el router y a la velocidad de
transferencia de nuestro sd-mod, se aumentan la mayoría de los tiempos
de espera en php para evitar “muertes” de procesos indeseadas.

max_input_time = 6000   ; Maximum amount of time each script may spend
parsing request data
memory_limit = 256M        ; Maximum amount of memory a script may
consume (8MB)
// También hemos habilitado un archivo de memoria swap, para que no se
desborde la ram del router, detallado en el punto 3.5.4

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Error handling and logging ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; error_reporting is a bit-field.  Or each number up to get desired
error
; reporting level
; E_ALL                  - All errors and warnings (doesn't include
E_STRICT)
; E_ERROR           - fatal run-time errors
; E_WARNING         - run-time warnings (non-fatal errors)
; E_PARSE           - compile-time parse errors
; E_NOTICE          - run-time notices (these are warnings which often
result
;                     from a bug in your code, but it's possible that
it was
;                          intentional (e.g., using an uninitialized
variable and
;                            relying on the fact it's automatically
initialized to an
;                     empty string)
; E_STRICT            - run-time notices, enable to have PHP suggest
changes
;                           to your code which will ensure the best
interoperability
;                     and forward compatibility of your code
; E_CORE_ERROR        - fatal errors that occur during PHP's initial
startup
; E_CORE_WARNING     - warnings (non-fatal errors) that occur during
PHP's
;                     initial startup
; E_COMPILE_ERROR   - fatal compile-time errors
; E_COMPILE_WARNING - compile-time warnings (non-fatal errors)
; E_USER_ERROR      - user-generated error message
; E_USER_WARNING    - user-generated warning message
; E_USER_NOTICE     - user-generated notice message
;
; Examples:
;
```

```
;     - Show all errors, except for notices and coding standards
warnings
;
;error_reporting = E_ALL & ~E_NOTICE & ~E_STRICT
;
;   - Show all errors, except for notices
;
;error_reporting = E_ALL & ~E_NOTICE
;
;   - Show only errors
;
;error_reporting = E_COMPILE_ERROR|E_ERROR|E_CORE_ERROR
;
;   - Show all errors except for notices and coding standards warnings
;
```
error_reporting = E_ALL & ~E_NOTICE & ~E_STRICT //Habilitamos para control de errores

```
; Print out errors (as a part of the output).  For production web
sites,
; you're strongly encouraged to turn this feature off, and use error
logging
; instead (see below).  Keeping display_errors enabled on a production
web site
; may reveal security information to end users, such as file paths on
your Web
; server, your database schema or other information.
```
display_errors = On

```
; Even when display_errors is on, errors that occur during PHP's
startup
; sequence are not displayed.  It's strongly recommended to keep
; display_startup_errors off, except for when debugging.
display_startup_errors = Off

; Log errors into a log file (server-specific log, stderr, or
error_log (below))
; As stated above, you're strongly advised to use error logging in
place of
; error displaying on production web sites.
log_errors = On

; Set maximum length of log_errors. In error_log information about the
source is
; added. The default is 1024 and 0 allows to not apply any maximum
length at all.
log_errors_max_len = 1024

; Do not log repeated messages. Repeated errors must occur in same
file on same
; line until ignore_repeated_source is set true.
ignore_repeated_errors = Off

; Ignore source of message when ignoring repeated messages. When this
setting
; is On you will not log errors with repeated messages from different
files or
; sourcelines.
ignore_repeated_source = Off
```

```
; If this parameter is set to Off, then memory leaks will not be shown
(on
; stdout or in the log). This has only effect in a debug compile, and
if
; error reporting includes E_WARNING in the allowed list
report_memleaks = On

; Store the last error/warning message in $php_errormsg (boolean).
track_errors = Off

; Disable the inclusion of HTML tags in error messages.
; Note: Never use this feature for production boxes.
;html_errors = Off

; If html_errors is set On PHP produces clickable error messages that
direct
; to a page describing the error or function causing the error in
detail.
;   You   can   download   a   copy   of   the   PHP   manual   from
http://www.php.net/docs.php
; and change docref_root to the base URL of your local copy including
the
; leading '/'. You must also specify the file extension being used
including
; the dot.
; Note: Never use this feature for production boxes.
;docref_root = "/phpmanual/"
;docref_ext = .html

; String to output before an error message.
;error_prepend_string = "<font color=ff0000>"

; String to output after an error message.
;error_append_string = "</font>"

; Log errors to specified file.
error_log   =   /mnt/sd/var/log/php-scripts.log   //Archivo   donde   php
guardará su log de errores.

; Log errors to syslog (Event Log on NT, not valid in Windows 95).
;error_log = syslog


;;;;;;;;;;;;;;;;;
; Data Handling ;
;;;;;;;;;;;;;;;;;
;
; Note - track_vars is ALWAYS enabled as of PHP 4.0.3

; The separator used in PHP generated URLs to separate arguments.
; Default is "&".
;arg_separator.output = "&amp;"

; List of separator(s) used by PHP to parse input URLs into variables.
; Default is "&".
; NOTE: Every character in this directive is considered as separator!
;arg_separator.input = ";&"

; This directive describes the order in which PHP registers GET, POST,
Cookie,
```

```
; Environment and Built-in variables (G, P, C, E & S respectively,
often
; referred to as EGPCS or GPC).  Registration is done from left to
right, newer
; values override older values.
variables_order = "EGPCS"

; Whether or not to register the EGPCS variables as global variables.
You may
; want to turn this off if you don't want to clutter your scripts'
global scope
; with user data.  This makes most sense when coupled with track_vars
- in which
;  case you can access all of the GPC variables through the
$HTTP_*_VARS[],
; variables.
;
; You should do your best to write your scripts so that they do not
require
;  register_globals to be on;  Using form variables as globals can
easily lead
; to possible security problems, if the code is not very well thought
of.
register_globals = Off

; Whether or not to register the old-style input arrays, HTTP_GET_VARS
;  and friends.  If you're not using them, it's recommended to turn
them off,
; for performance reasons.
register_long_arrays = On

; This directive tells PHP whether to declare the argv&argc variables
(that
;  would contain the GET information).  If you don't use these
variables, you
; should turn it off for increased performance.
register_argc_argv = On

; Maximum size of POST data that PHP will accept.
post_max_size = 64M

; Magic quotes
;

; Magic quotes for incoming GET/POST/Cookie data.
magic_quotes_gpc = On

; Magic quotes for runtime-generated data, e.g. data from SQL, from
exec(), etc.
magic_quotes_runtime = Off

; Use Sybase-style magic quotes (escape ' with '' instead of \').
magic_quotes_sybase = Off

; Automatically add files before or after any PHP document.
auto_prepend_file =
auto_append_file =

; As of 4.0b4, PHP always outputs a character encoding by default in
; the Content-type: header.  To disable sending of the charset, simply
; set it to be empty.
```

```
;
; PHP's built-in default is text/html
default_mimetype = "text/html"
#;default_charset = "iso-8859-1"
default_charset = "UTF-8"
; Always populate the $HTTP_RAW_POST_DATA variable.
;always_populate_raw_post_data = On


;;;;;;;;;;;;;;;;;;;;;;;;
; Paths and Directories ;
;;;;;;;;;;;;;;;;;;;;;;;;

; UNIX: "/path1:/path2"
;include_path = ".:/php/includes"
;
; Windows: "\path1;\path2"
;include_path = ".;c:\php\includes"

include_path                                                    =
".:/usr/lib/php:/mnt/sd/usr/lib/php:/mnt/sd/www/moodle/moodle:/mnt/sd/
:/mnt/sd/www/:/mnt/sd/www/moodle" //El raíz a partir del cual están
los archivos .php

; The root of the PHP pages, used only if nonempty.
; if PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc_root
; if you are running php as a CGI under any web server (other than
IIS)
; see documentation for security issues.  The alternate is to use the
; cgi.force_redirect configuration below
doc_root = /mnt/sd/www

; The directory under which PHP opens the script using /~username used
only
; if nonempty.
user_dir =/mnt/sd/usr/bin

; Directory in which the loadable extensions (modules) reside.
extension_dir = "/mnt/sd/usr/lib/php"

; Whether or not to enable the dl() function.  The dl() function does
NOT work
; properly in multithreaded servers, such as IIS or Zeus, and is
automatically
; disabled on them.
enable_dl = On

; cgi.force_redirect is necessary to provide security running PHP as a
CGI under
; most web servers.  Left undefined, PHP turns this on by default.
You can
; turn it off here AT YOUR OWN RISK
; **You CAN safely turn this off for IIS, in fact, you MUST.**
; cgi.force_redirect = 1

; if cgi.nph is enabled it will force cgi to always sent Status: 200
with
; every request.
; cgi.nph = 1
```

```
; if cgi.force_redirect is turned on, and you are not running under
Apache or Netscape
; (iPlanet) web servers, you MAY need to set an environment variable
name that PHP
; will look for to know it is OK to continue execution.  Setting this
variable MAY
; cause security issues, KNOW WHAT YOU ARE DOING FIRST.
; cgi.redirect_status_env = ;

; FastCGI under IIS (on WINNT based OS) supports the ability to
impersonate
; security tokens of the calling client.  This allows IIS to define
the
; security context that the request runs under.  mod_fastcgi under
Apache
; does not currently support this feature (03/17/2002)
; Set to 1 if running under IIS.  Default is zero.
; fastcgi.impersonate = 1;

; cgi.rfc2616_headers configuration option tells PHP what type of
headers to
; use when sending HTTP response code. If it's set 0 PHP sends Status:
header that
; is supported by Apache. When this option is set to 1 PHP will send
; RFC2616 compliant header.
; Default is zero.
;cgi.rfc2616_headers = 0


;;;;;;;;;;;;;;;;
; File Uploads ;
;;;;;;;;;;;;;;;;

; Whether to allow HTTP file uploads.
file_uploads = On

; Temporary directory for HTTP uploaded files (will use system default
if not
; specified).
upload_tmp_dir = /tmp

; Maximum allowed size for uploaded files.
upload_max_filesize = 2M


;;;;;;;;;;;;;;;;;;
; Fopen wrappers ;
;;;;;;;;;;;;;;;;;;

; Whether to allow the treatment of URLs (like http:// or ftp://) as
files.
allow_url_fopen = On

; Define the anonymous ftp password (your email address)
;from="john@doe.com"

; Define the User-Agent string
; user_agent="PHP"

; Default timeout for socket based streams (seconds)
default_socket_timeout =360
```

```
; If your scripts have to deal with files from Macintosh systems,
; or you are running on a Mac and need to deal with files from
; unix or win32 systems, setting this flag will cause PHP to
; automatically detect the EOL character in those files so that
; fgets() and file() will work regardless of the source of the file.
auto_detect_line_endings = On
#auto_detect ANTES OFF


;;;;;;;;;;;;;;;;;;;;;
; Dynamic Extensions ;
;;;;;;;;;;;;;;;;;;;;;
;
;  If you wish to have an extension loaded automatically, use the
following
; syntax:
;
;   extension=modulename.extension
;
; For example, on Windows:
;
;   extension=msql.dll
;
; ... or under UNIX:
;
;   extension=msql.so
;
;  Note that it should be the name of the module only; no directory
information
; needs to go here.  Specify the location of the extension with the
; extension_dir directive above.


;Windows Extensions
;Note that ODBC support is built in, so no dll is needed for it.
;

;extension=ftp.so
extension=curl.so
extension=gd.so
extension=mysql.so
extension=fastcgi.so
extension=rewrite.so
extension=accesslog.so
extension=pcre.so
extension=session.so
;extension=sockets.so
extension=xml.so
extension=sqlite.so
extension=pdo_sqlite.so
extension=pdo.so
extension=pgsql.so
extension=openssl.so

//Habilitadas algunas extensiones necesarias para el funcionamiento de
wordpress y postgresql

;;;;;;;;;;;;;;;;;;;
; Module Settings ;
;;;;;;;;;;;;;;;;;;;
```

```
[SQL]
sql.safe_mode = Off

[Session]
; Handler used to store/retrieve data.
session.save_handler = files

; Argument passed to save_handler.  In the case of files, this is the
path
; where data files are stored. Note: Windows users have to change this
; variable in order to use PHP's session functions.
;
; As of PHP 4.0.1, you can define the path as:
;
;     session.save_path = "N;/path"
;
; where N is an integer.  Instead of storing all the session files in
; /path, what this will do is use subdirectories N-levels deep, and
; store the session data in those directories.  This is useful if you
; or your OS have problems with lots of files in one directory, and is
; a more efficient layout for servers that handle lots of sessions.
;
; NOTE 1: PHP will not create this directory structure automatically.
;            You can use the script in the ext/session dir for that
purpose.
; NOTE 2: See the section on garbage collection below if you choose to
;         use subdirectories for session storage
;
; The file storage module creates files using mode 600 by default.
; You can change that by using
;
;     session.save_path = "N;MODE;/path"
;
; where MODE is the octal representation of the mode. Note that this
; does not overwrite the process's umask.
#session.save_path = "/mnt/sd/usr/lib/php/session"
session.save_path = "/tmp" //Aquí es donde guardamos las sesiones,
éste directorio se borra cada vez que reiniciamos el sistema.

; Whether to use cookies.
//session.use_cookies = 1
session.use_cookies = 0 //Finalmente, se decidió implementar unas
sesiones “cookie-less”, pues los scripts morían en las cargas, se
pusiera el tiempo que se pusiera.

; This option enables administrators to make their users invulnerable
to
; attacks which involve passing session ids in URLs; defaults to 0.
; session.use_only_cookies = 1

; Name of the session (used as cookie name).
session.name = PHPSESSID

; Initialize session on request startup.
//session.auto_start = 0
session.auto_start = 0
; Lifetime in seconds of cookie or, if 0, until browser is restarted.
session.cookie_lifetime = 1600 //Tiempo de vida de las cookies en
segundos.
```

```
; The path for which the cookie is valid.
//session.cookie_path = /mnt/sd/www
```
session.cookie_path = /mnt/sd/www/moodle //Ruta donde guardamos las cookies
```
; The domain for which the cookie is valid.
```
session.cookie_domain ='192.168.1.2' //Cookies válidas para la dirección ip del router.
```
; Handler used to serialize data.  php is the standard serializer of PHP.
session.serialize_handler = php

; Define the probability that the 'garbage collection' process is started
; on every session initialization.
; The probability is calculated by using gc_probability/gc_divisor,
; e.g. 1/100 means there is a 1% chance that the GC process starts
; on each request.

session.gc_probability = 1
session.gc_divisor     = 100

; After this number of seconds, stored data will be seen as 'garbage' and
; cleaned up by the garbage collection process.
session.gc_maxlifetime = 1600

; NOTE: If you are using the subdirectory option for storing session files
;         (see session.save_path above), then garbage collection does *not*
;       happen automatically.  You will need to do your own garbage
;        collection through a shell script, cron entry, or some other method.
;       For example, the following script would is the equivalent of
;          setting session.gc_maxlifetime to 1440 (1440 seconds = 24 minutes):
;          cd /path/to/sessions; find -cmin +24 | xargs rm

; PHP 4.2 and less have an undocumented feature/bug that allows you to
;  to initialize a session variable in the global scope, albeit register_globals
; is disabled.  PHP 4.3 and later will warn you, if this feature is used.
; You can disable the feature and the warning separately. At this time,
; the warning is only displayed, if bug_compat_42 is enabled.

session.bug_compat_42 = 1
session.bug_compat_warn = 0

; Check HTTP Referer to invalidate externally stored URLs containing ids.
; HTTP_REFERER has to contain this substring for the session to be
; considered as valid.
session.referer_check =

; How many bytes to read from the file.
session.entropy_length = 0

; Specified here to create the session id.
```

```
session.entropy_file =

;session.entropy_length = 16

;session.entropy_file = /dev/urandom

; Set to {nocache,private,public,} to determine HTTP caching aspects
; or leave this empty to avoid sending anti-caching headers.
session.cache_limiter = nocache

; Document expires after n minutes.
session.cache_expire = 380 //A los 380 minutos caduca la sesión.

; trans sid support is disabled by default.
; Use of trans sid may risk your users security.
; Use this option with caution.
; - User may send URL contains active session ID
;   to other person via. email/irc/etc.
; - URL that contains active session ID may be stored
;   in publically accessible computer.
; - User may access your site with the same session ID
;   always using URL stored in browser's history or bookmarks.
//session.use_trans_sid = 0
session.use_trans_sid = 1;
; Select a hash function
; 0: MD5   (128 bits)
; 1: SHA-1 (160 bits)
session.hash_function = 0

; Define how many bits are stored in each character when converting
; the binary hash data to something readable.
;
; 4 bits: 0-9, a-f
; 5 bits: 0-9, a-v
; 6 bits: 0-9, a-z, A-Z, "-", ","
session.hash_bits_per_character = 4

; The URL rewriter will look for URLs in a defined set of HTML tags.
; form/fieldset are special; if you include them here, the rewriter
will
; add a hidden <input> field with the info which is otherwise appended
; to URLs.  If you want XHTML conformity, remove the form entry.
; Note that all valid entries require a "=", even if no value follows.
url_rewriter.tags                                                    =
"a=href,area=href,frame=src,input=src,form=,fieldset="


[Assertion]
; Assert(expr); active by default.
;assert.active = On

; Issue a PHP warning for each failed assertion.
;assert.warning = On

; Don't bail out by default.
;assert.bail = Off

; User-function to be called if an assertion fails.
;assert.callback = 0
```

```
; Eval the expression with current error_reporting().  Set to true if
you want
; error_reporting(0) around the eval().
;assert.quiet_eval = 0




[exif]
; Exif UNICODE user comments are handled as UCS-2BE/UCS-2LE and JIS as
JIS.
; With mbstring support this will automatically be converted into the
encoding
;    given    by    corresponding    encode    setting.    When    empty
mbstring.internal_encoding
;  is  used.  For  the  decode  settings  you  can  distinguish  between
motorola and
; intel byte order. A decode setting cannot be empty.
;exif.encode_unicode = ISO-8859-15
;exif.decode_unicode_motorola = UCS-2BE
;exif.decode_unicode_intel    = UCS-2LE
;exif.encode_jis =
;exif.decode_jis_motorola = JIS
;exif.decode_jis_intel    = JIS
```

Una vez configurado el archivo php.ini, pasamos a comprobar si carga los módulos correctamente mediante el comando:

php-cgi -m

Esto no debería lanzar ningún error y debería mostrar los módulos inicializados como en la *figura 13*:

```
root@OpenWrt:/mnt/sd/etc# php-cgi -m
[PHP Modules]
apc
cgi
curl
date
exif
gd
hash
json
openssl
pcre
PDO
pdo_sqlite
pgsql
posix
Reflection
session
sockets
SQLite
standard
xml
zlib

[Zend Modules]
```

Figura 13. Resultado de ejecutar el comando php-cgi –m.

Ahora vamos a comprobar a través del navegador, si funciona correctamente.
Para ello se crea un archivo llamado php-info.php con el siguiente contenido:

```
<?php phpinfo(); ?>
```

Y se guarda en la carpeta raíz a partir de la cual empieza a leer el servidor lighttpd (mnt/sd/www)

Una vez creado, (y con el servidor lighttpd arrancado), escribimos en la barra de direcciones del navegador:

http://192.168.1.2/phpinfo.php

Y nos cargará la página .php con toda la información que hemos configurado en el php.ini (Ver *figura 14*).

Figura 14. Parte de la página cargada phpinfo.php

En éste punto nuestro servidor ya puede leer páginas web con contenido dinámico, pero necesitaremos habilitar memoria extra por medio de un archivo swap en nuestra tarjeta sd, para evitar cuelgues y desbordes de memoria de nuestro router.

4.5.4 Habilitando Memoria Extra (SWAP)

Éste tipo de memoria empieza a ocuparse una vez que la memoria del router (16MB) se ha llenado. El sistema operativo reserva, de manera automática, una pequeña parte de su ram para seguir funcionando y empieza a volcar memoria a nuestro archivo swap (192MB)

Para poder utilizar los comandos para crear y manipular archivos swap, deberemos instalar el package “swap_utils” del repositorio:

opkg –d sd install swap_utils

La creación de la swap se lleva a cabo mediante los siguientes comandos:

dd if=/dev/zero of=/mnt/sd/myswap.swp bs = 1024 count = 196608

De ésta manera se crea un archivo swap en /mnt/sd de longitud 192MB (192x1024=196608 bits)

A continuación, con los siguientes comandos, habilitamos el archivo swap y le hacemos saber al sistema que puede utilizarlo como memoria swap:

mkswap /mnt/sd/myswap.swp
swapon /mnt/sd/myswap.swp

De ésta manera quedaría habilitada la memoria extra, usando el comando *free*, podemos comprobar que tenemos correctamente configurado éste espacio extra (ver figura 15).

```
root@OpenWrt:/mnt/sd/etc# free
             total         used         free       shared      buffers
  Mem:        14328        12356         1972            0          464
 Swap:       191992           40       191952
Total:       206320        12396       193924
root@OpenWrt:/mnt/sd/etc#
```

Figura 15. Resultado de ejecutar el comando *free*

Se habilita la asignación de la memoria swap tras el reinicio añadiendo la siguiente línea al archivo /etc/fstab:

/mnt/sd/myswap.swp swap swap defaults 0 0

4.6 Configurando la Base de Datos (PostgreSQL)

4.6.1. Introducción

Dado que vamos a utilizar un sistema de gestor de contenidos (Wordpress) necesitaremos crear una base de datos para almacenar los contenidos del blog.
Para éste proyecto, se ha utilizado el paquete PostgreSQL para openwrt, se eligió Postgresql sobre MySQL por la sencilla razón de que el servidor MySQL consumía una enorme cantidad de recursos, PostgreSQL, también consume una buena cantidad de recursos, pero no de una manera tan exagerada.

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, bajo licencia BSD (*Berkeley Software Distribution*).

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una empresa y/o persona, sino que es dirigido por una comunidad de desarrolladores que trabajan de forma desinteresada, altruista, libre y/o apoyados por organizadores comerciales. Dicha comunidad es denominada el PGDG (*PostgreSQL Global Development Group*).

4.6.2 Creación del Usuario Postgres

PostgreSQL necesita ejecutarse como usuario y nosotros estamos ejecutando todo como root, por lo que crearemos un usuario sin permisos de superusuario llamado postgres para éste caso, para ello, nos conectados como root en el terminal y dentro del router escribiremos:

echo "postgres::1000:1000:postgres:/home/postgres:/bin/ash" >> /etc/passwd*
echo "postgres:x:1000: " >> /etc/group
mkdir /home/postgres
chown postgres.postgres:/home/postgres
passwd 123456

Se nos creará un nuevo usuario llamado postgres con password 123456.
Para la instalación y configuración, conectaremos al router mediante:

ssh postgres@192.168.1.2

Tendremos 2 terminales abiertos, uno con el usuario root (para la instalación de paquetes y demás) y otro con el usuario postgres para la configuración de la base de datos.

4.6.3 Instalación y Configuración

La instalación de los paquetes necesarios para hacer funcionar PostgreSQL se realiza mediante la siguiente instrucción (desde el terminal root):

opkg –d sd install php5-mod-pgsql pgsql-cli pgsql-server

El sistema se encargará de instalar las dependencias necesarias.
Para la configuración, modificaremos el archivo postgresql situado en /mnt/sd/etc/config/postgresql y cambiaremos la ruta donde se van a guardar los datos y la ruta donde se guardará el log de errores. En nuestro caso las líneas con # eran las que venían por defecto y las que no la llevan son las líneas introducidas por el autor:

```
#option PGDATA  /var/postgresql/data
#option PGLOG   /var/postgresql/data/postgresql.log
  option PGDATA /mnt/sd/www/postgresql
  option PGLOG /mnt/sd/www/postgresql/postgresql.log
```

Crearemos la carpeta postgresql y le daremos permisos de lectura, escritura y ejecución al usuario postgres sobre ésa carpeta:

mkdir /mnt/sd/www/postgresql
chown postgres:postgres postgresql

Vamos a configurar la base de datos, para ello primero ejecutamos la siguiente instrucción (desde el terminal con el usuario postgres):

LL_COLLATE=”c” initdb –-pwprompt –D /mnt/sd/www/postgresql

Tras unos minutos de configuración, nos habrá creado la carpeta /mnt/sd/www/postgresql donde se almacenarán todas las tablas e información de la base de datos (ver figura 16).

```
postgres@OpenWrt:~$ LL_COLLATE="c" initdb --pwprompt -D /mnt/sd/www/postgresql
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale C.
The default database encoding has accordingly been set to SQL_ASCII.
The default text search configuration will be set to "english".

fixing permissions on existing directory /mnt/sd/www/postgresql ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers/max_fsm_pages ... 24MB/153600
creating configuration files ... ok
creating template1 database in /mnt/sd/www/postgresql/base/1 ... ok
initializing pg_authid ... ok
Enter new superuser password:
Enter it again:
setting password ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
creating dictionaries ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    postgres -D /mnt/sd/www/postgresql
or
    pg_ctl -D /mnt/sd/www/postgresql -l logfile start

postgres@OpenWrt:~$
```

Figura 16. Configuración de la base de datos

Una vez terminado podremos inicializar la base de datos con la siguiente instrucción:

postgres –D /mnt/sd/www/postgres

Y la base de datos quedará lista para recibir peticiones. Si conectamos por terminal (con el usuario postgres) con la base de datos y hacemos \l nos aparecerán las bases de datos disponibles y con el formato de caracteres de cada una de ellas (Ver figura 17).

```
postgres@OpenWrt:~$ psql -d postgres -U postgres
Welcome to psql 8.3.3, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

postgres=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+-----------
 postgres  | postgres | SQL_ASCII
 template0 | postgres | SQL_ASCII
 template1 | postgres | SQL_ASCII
(3 rows)

postgres=#
```

Figura 17. Lista de bases de datos

Podemos observar como los formatos de codificación son SQL_ASCII, para evitar problemas de compatibilidad con wordpress, crearemos una nueva base de datos llamada wordpress, con formato UTF8 (UNICODE). Ejecutamos la instrucción:

```
postgres=# CREATE DATABASE wordpress WITH ENCODING 'UNICODE';
```

Tras ésta instrucción, nos aparecerá una nueva base de datos, cuyo propietario, de momento es postgres (ver figura 18)

```
postgres=# CREATE DATABASE wordpress WITH ENCODING 'UNICODE';
CREATE DATABASE
postgres=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+-----------
 postgres  | postgres | SQL_ASCII
 template0 | postgres | SQL_ASCII
 template1 | postgres | SQL_ASCII
 wordpress | postgres | UTF8
(4 rows)

postgres=#
```

Figura 18. Lista de bases de datos

Vamos a crear un nuevo usuario llamado “wordpressuser” con password “123456”, y le asignamos como propietario de ésa base de datos, lo haremos con las dos siguientes líneas:

```
postgres=# CREATE ROLE wordpressuser WITH PASSWORD '123456';
postgres=# ALTER DATABASE wordpress OWNER TO wordpressuser;
```

De ésta manera tendremos la nueva base de datos wordpress, con codificación UTF8, usuario wordpressuser, contraseña 123456, datos que tendremos que introducir en el archivo de configuración de wordpress para la conexión con la base de datos.

```
postgres=# CREATE ROLE wordpressuser WITH PASSWORD '123456';
CREATE ROLE
postgres=# ALTER DATABASE wordpress OWNER TO wordpressuser;
ALTER DATABASE
postgres=# \l
           List of databases
   Name    |     Owner     | Encoding
-----------+---------------+-----------
 postgres  | postgres      | SQL_ASCII
 template0 | postgres      | SQL_ASCII
 template1 | postgres      | SQL_ASCII
 wordpress | wordpressuser | UTF8
(4 rows)

postgres=#
```

Figura 19. Lista de bases de datos

Deberemos permitir acceso al usuario wordpressuser

```
postgres=# ALTER ROLE wordpressuser LOGIN;
```

También se deberá regular el acceso mediante password en el archivo /mnt/sd/www/wordpress/pb_hba.conf, mediante “local all all password”, lo que da acceso a cualquier usuario para cualquier base de datos *siempre mediante password*.

```
# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD

# "local" is for Unix domain socket connections only
#local   all    all     md5
#local  all     all     trust
local   all     all     password
```

Figura 20. Archivo pg_hba.conf

4.7 Instalación de Wordpress

4.7.1. ¿Qué es Wordpress y Cual es su Cometido en Éste PFC?

Wordpress es un sistema de gestión de contenido (*Control Management System, CMS)* dedicado a la creación de blogs. La idea es que los docentes pueden subir su contenido periódicamente al blog y mantener una enseñanza virtual de calidad.

WordPress está desarrollado en PHP y MySQL, bajo licencia GPL y código modificable, tiene como fundador a Matt Mullenweg. WordPress fue creado a partir del desaparecido **b2/cafelog** y se ha convertido junto a Movable Type en el CMS más popular de la blogosferay en el más popular con respecto a cualquier otro CMS de aplicación general. Las causas de su enorme crecimiento son, entre otras, su licencia, su facilidad de uso y sus características como gestor de contenidos.

Otro motivo a considerar sobre su éxito y extensión, es la enorme comunidad de desarrolladores y diseñadores, que se encargan de desarrollarlo en general o crear *plugins* y temas para la comunidad, siendo usado por el 14.7% de todos los sitios existentes en internet.

4.7.2 Instalación y configuración

El primer paso es descargar la última versión de wordpress desde la página oficial, en éste caso, hemos instalado la versión 3.3.2 en español.
Desde el terminal, se navega al directorio donde está guardado el paquete descargado y se envía por scp a nuestro router:

scp wordpress-3.3.2-es_ES.zip root@192.168.1.2:/mnt/sd/www/

Una vez dentro lo descomprimimos en la carpeta /mnt/sd/www/wordpress, abrimos el archivo /mnt/sd/www/wordpress/wp-config-sample.php, lo editamos para añadir las opciones que se ajusten a nuestro equipo y lo guardamos renombrado como wp-config.php.
Éste es el código fuente el archivo y en negrita lo añadido por el autor:

```
<?php
/**
 * The base configurations of the WordPress.
 *
 * This file has the following configurations: MySQL settings, Table Prefix,
 * Secret Keys, WordPress Language, and ABSPATH. You can find more information
```

```
 * by visiting {@link http://codex.wordpress.org/Editing_wp-config.php
Editing

 * wp-config.php} Codex page. You can get the MySQL settings from your
web host.

 *

 * This file is used by the wp-config.php creation script during the

 * installation. You don't have to use the web site, you can just copy
this file

 * to "wp-config.php" and fill in the values.

 *

 * @package WordPress

 */


// ** MySQL settings - You can get this info from your web host ** //

/** The name of the database for WordPress */

define('DB_NAME', 'wordpress');


/** MySQL database username */

define('DB_USER', 'wordpressuser');


/** MySQL database password */

define('DB_PASSWORD', '123456');


/** MySQL hostname */

define('DB_HOST', 'localhost');


/** Database Charset to use in creating database tables. */

define('DB_CHARSET', 'utf8');


/** The Database Collate type. Don't change this if in doubt. */

define('DB_COLLATE', '');
```

```
/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');

/**#@-*/

/**
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each a unique
 * prefix. Only numbers, letters, and underscores please!
 */
$table_prefix  = 'wp_';

/**
```

```
 * WordPress Localized Language, defaults to English.

 *

 * Change this to localize WordPress. A corresponding MO file for the
chosen

 * language must be installed to wp-content/languages. For example,
install

 * de_DE.mo to wp-content/languages and set WPLANG to 'de_DE' to
enable German

 * language support.

 */

define('WPLANG', 'es_ES');



/**

 * For developers: WordPress debugging mode.

 *

 * Change this to true to enable the display of notices during
development.

 * It is strongly recommended that plugin and theme developers use
WP_DEBUG

 * in their development environments.

 */

define('WP_DEBUG', false);



/* That's all, stop editing! Happy blogging. */



/** Absolute path to the WordPress directory. */

if ( !defined('ABSPATH') )

      define('ABSPATH', dirname(__FILE__) . '/');



/** Sets up WordPress vars and included files. */

require_once(ABSPATH . 'wp-settings.php');
```

Arrancamos la base de datos y el servidor lighttpd en caso de que no lo estuviera y escribimos en el navegador http://192.168.1.2/wordpress/wp-admin/install.php
(Importante tener cookies deshabilitadas)

Nos aparecerá una pantalla de bienvenida con los campos a rellenar (figura 21)

Figura 21. Instalación de wordpress

Rellenamos los campos con nuestra información (es sensible a mayúsculas y minúsculas).

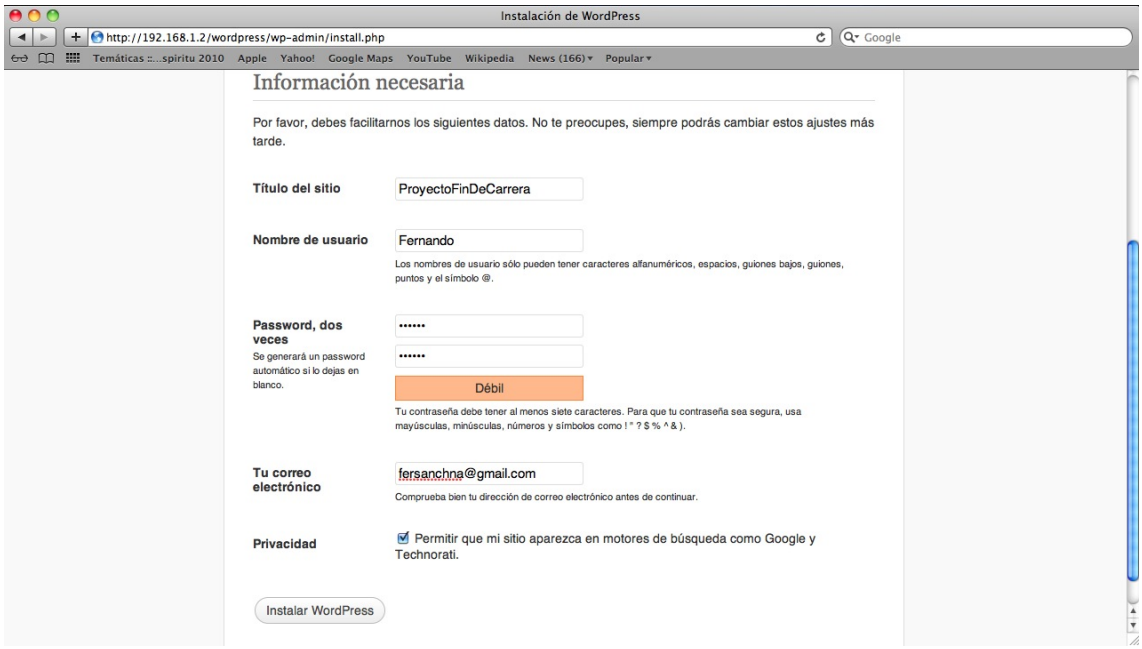

Figura 22. Instalación de wordpress

El mismo programa de instalación configura todas las tablas necesarias y los pocos minutos, queda todo instalado y listo para utilizar, sólo nos quedaría introducir el login y password y tendríamos acceso total a la configuración del blog.

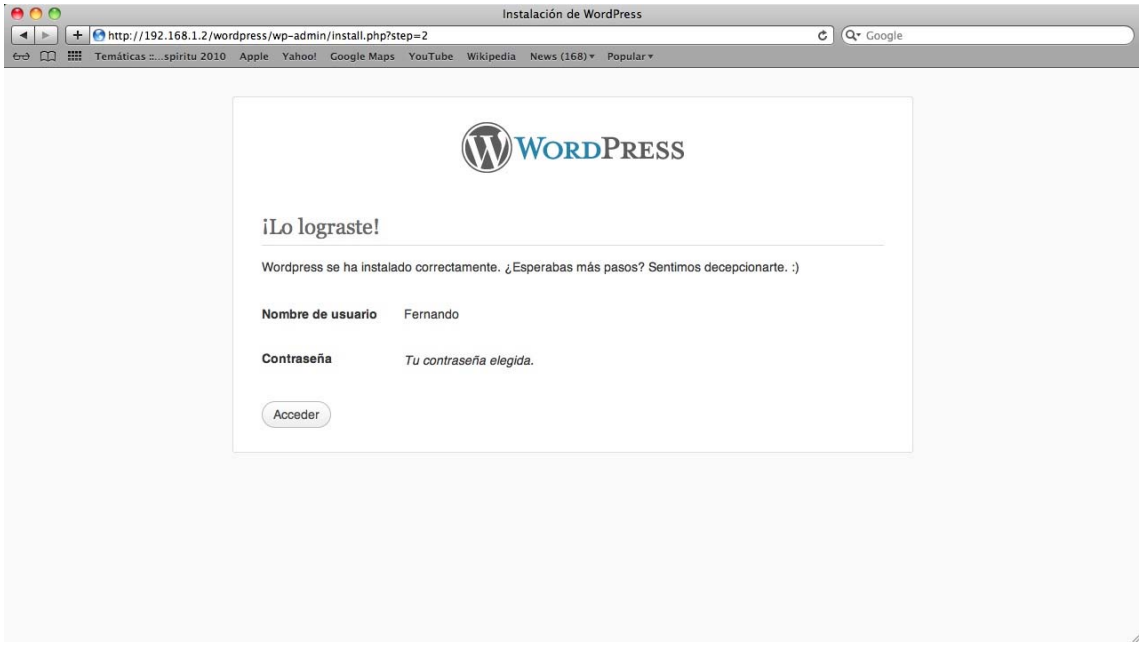

Figura 23. Fin de la instalación de wordpress

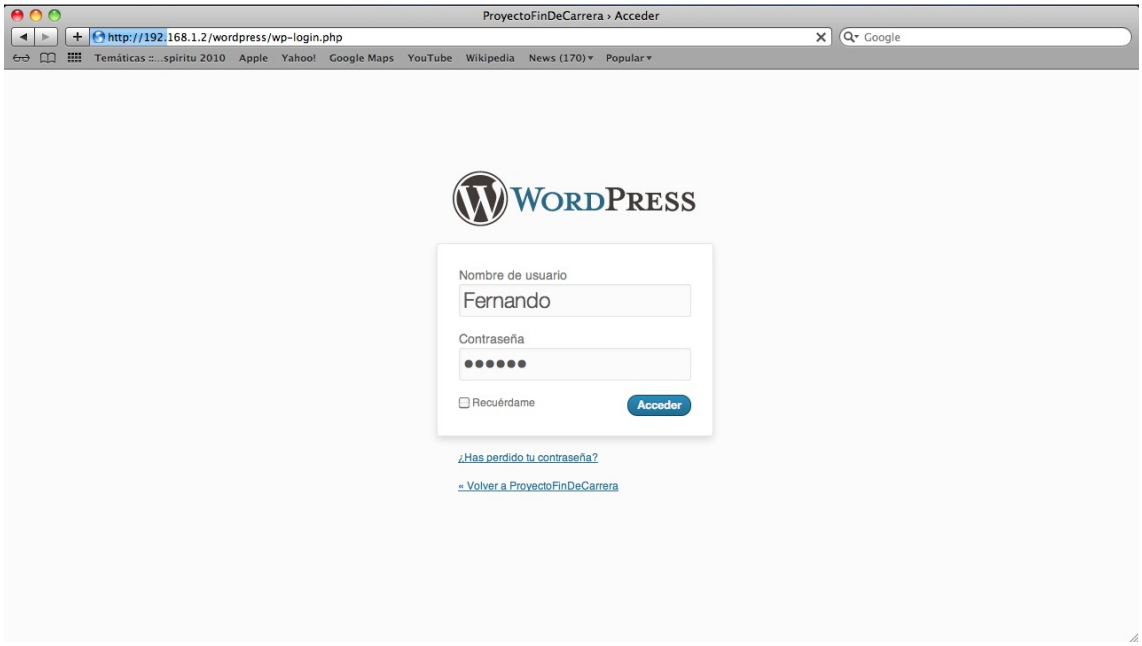

Figura 24. Pantalla de login

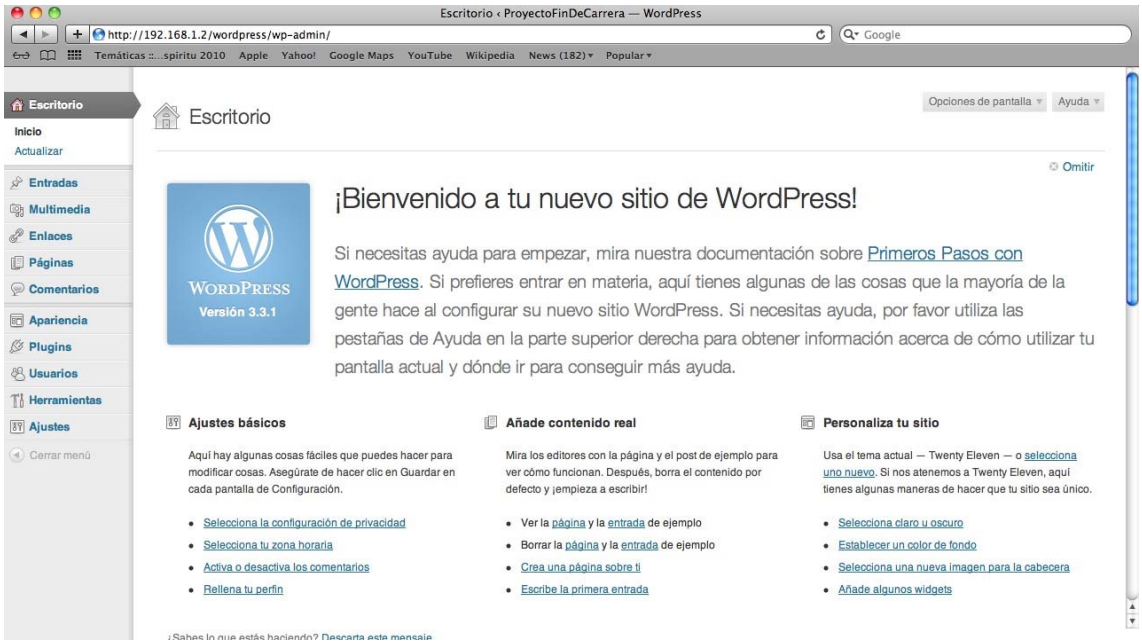

Figura 25. Pantalla de bienvenida

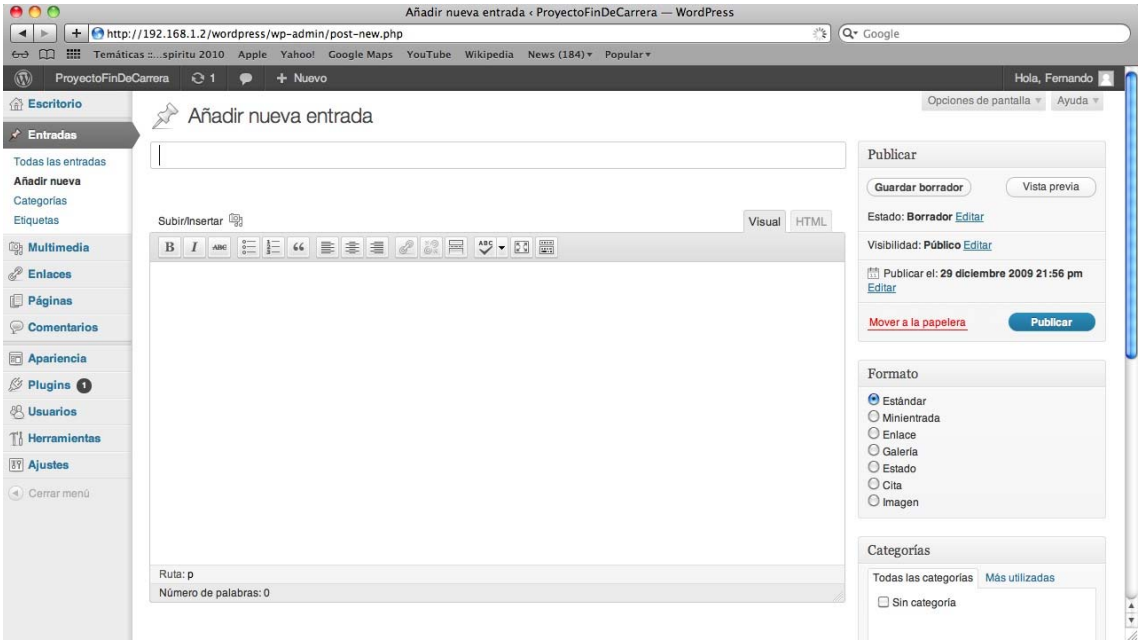

Figura 26. Pantalla para añadir nuevo post

5 CONCLUSIONES Y LINEAS FUTURAS

Finalmente se consiguieron llevar a cabo todos los objetivos, los cuales pasamos a detallar a continuación punto por punto:

- Se consiguió el aumento de la capacidad de almacenamiento del router instalando un slot lector de tarjetas SD, soldándolo a la placa y configurando un driver para leer las tarjetas.

- Se ha instalado openwrt con la distribución kamikaze 8.09.2, sin ningún tipo de problema, los repositorios de paquetes para ésta distribución eran los más variados en cuanto a paquetes y es por ello por lo que se decidió instalar ésta distribución.

- Se utilizó un servidor ligero http llamado “lighttpd”, rápido y eficiente, facilitó el servicio de páginas web en un router con recursos limitados.

- Se instaló un paquete precompilado de php del repositorio de kamikaze, ha sido la versión 5.2.6

- Se intentó instalar en primer lugar SQLITE, fue un proyecto que se inició en 2008 y finalmente no llegó a buen puerto y se abandonó, por lo que el proyecto estaba incompleto, llamó la atención del autor en un principio debido a que era el más ligero de los tres contemplados (SQLITE, MySQL y PostgreSQL). Se intentó MySQL, pero consumía demasiados recursos, tanto que hacía imposible trabajar con él. Finalmente se utilizó PostgreSQL, funciona algo lento debido a los recursos del sistema, pero era finalmente la opción más adecuada.

- Se probó en primer lugar instalar moodle, pero debido a los largos tiempos de espera, no se vió factible y se decidió instalar wordpress, manteniedo la idea de que los docentes puedan seguir subiendo su material y los alumnos acceder a los recursos.

Se podría retomar el proyecto (sería programar en C) el paquete SQLITE, aunque sería una línea de investigación para un equipo de personas, pues sería un proyecto de gran envergadura.
De ésta manera se optimizaría el rendimiento del sistema y del proyecto en sí.

6 BIBLIOGRAFÍA

-"*Linksys WRT54G Ultimate Hacking*" por Paul Asadoorian.
- www.dd-wrt.com
- http://es.wikipedia.org
- http://es.wordpress.org
- http://wiki.openwrt.org
- http://wiki.opdevel.com
- http://nxbot.blogspot.com.es/2007/06/router-hack.html