ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA


**Proyecto Fin de Carrera**

# Implementación OpenWrt del protocolo de comunicaciones W2LAN

AUTOR: Francisco Ortuño López
DIRECTOR: Francesc Burrull i Mestres

Junio / 2010


| | |
|---|---|
| **Autor** | Francisco Ortuño López |
| **E-mail del Autor** | f.ortuno.lopez@gmail.com |
| **Director(es)** | Francesc Burrull i Mestres |
| **E-mail del Director** | francesc.burrull@upct.es |
| **Título del PFC** | Implementación OpenWrt del protocolo de comunicaciones W2LAN |
| **Descriptores** | W2LAN, OpenWrt, Linksys WRT54GL |

**Resumen**

Partiendo del modelo SDL (Specification and Description Language) del protocolo de comunicaciones W2LAN se pretende obtener una implementación para OpenWrt. Dicha versión deberá ejecutarse en plataforma inalámbrica, en concreto sobre tarjetas Wi-Fi.

| | |
|---|---|
| **Titulación** | I.T. Telecomunicación Esp. Telemática |
| **Departamento** | Departamento de Tecnologías de la Información y las Comunicaciones |
| **Fecha de Presentación** | Junio - 2010 |

# Indice:

# 1. Objetivos.

El objetivo de este proyecto es portar el protocolo W2LAN, de nivel 2 de la capa OSI, a un sistema empotrado (router) con un sistema operativo basado en el kernel de Linux.

- Elección adecuada del sistema empotrado (router), debe ser un router que disponga de interfaz wireless y que soporte cambios de firmware.
- Elección del firmware adecuado el cual soporte la instalación de paquetes.
- Compilar el protocolo W2LAN para que se pueda ejecutar en la arquitectura del router.
- Crear un paquete con W2LAN para que se pueda instalar en cualquier router.
- Comprobar que el router es capaz de ejecutar el protocolo W2LAN.

## 2. Introducción.

El protocolo W2LAN (Wireless to LAN) es un protocolo de nivel de enlace que transforma desde el punto de vista de capas superiores una red 802.11x MANET en una red LAN Ethernet, manteniendo compatibilidad con otros protocolos ya bien establecidos y probados. Así, usando W2LAN cualquier par de terminales inalámbricos que pertenezcan a la misma LAN se verán entre ellos - visibilidad total-, siempre que al menos exista una ruta entre ellos (característica mínima necesaria en cualquier red).

Para poder crear una red en la que funcione W2LAN, necesitamos que este protocolo esté presente en todos los dispositivos que forman dicha red: router, PC, TPV, PDA..., por lo que el objetivo de este proyecto es portar el protocolo W2LAN a un router con una distribución de Linux.

Para conseguir el objetivo de que W2LAN se ejecute en un router, se parte de los siguientes puntos:

1. Analizar los diferentes SO empotrados que se pueden utilizar.
2. Seleccionar un SO empotrado y familiarizarse con las herramientas de desarrollo que nos proporciona.
3. Una vez realizados los puntos anteriores compilar w2lan para que pueda ser utilizado por el router.

Se debe crear un paquete con el protocolo w2lan para que sea sencilla su instalación en el router.

Una vez compilado e instalado, habrá que comprobar que W2LAN es capar de comunicarse con otros elementos de la red.

# 3. Posibles soluciones del proyecto.

## 3.1. Selección del router.

Lo primero que debemos hacer es seleccionar un router que nos permita cambiarle el firmware para poder instalarle uno que contenga una distribución de Linux, el router que seleccionaremos será el Linksys WRT54GL, ya que es un modelo económico, dispone de un puerto WAN, un switch Ethernet, una interfaz wireless, además permite la instalación de Linux y es altamente configurable, tanto a nivel de hardware como de software.


**Imagen 3.1: Router Linksys WRT54GL**

## 3.2. Selección del sistema operativo del router.

Una vez seleccionado el router que se utilizará, se procederá a seleccionar el SO basado en Linux que le instalaremos al router, en este apartado disponemos de numerosas opciones de las cuales solo nos centraremos en las que dispongan de licencia GNU GENERAL PUBLIC LICENSE y que tengan una comunidad de desarrolladores que garanticen la continuidad del proyecto.

A continuación se pasa a enumerar las distintas opciones que podemos utilizar:

- ***DD-WRT:*** es un firmware desarrollado bajo licencia GPL para una gran variedad de routers inalámbricos (IEEE 802.11 a/b/g/h/n) basados en un chip broadcom o atheros.


**Imagen 3.2: Logo DD-Wrt**

El firmware es mantenido por Brain Slayer y está alojado en www.dd-wrt.com. Las primeras versiones de DD-WRT se basaron en el firmware Alchemy de Sveasoft Inc., que a su vez estaba basado en el firmware original de Linksys con licencia GPL y en una serie de proyectos de código abierto. DD-WRT fue creado directamente desde el software de Sveasoft, pero la decisión de comenzar a cobrar por su firmware, cerró la puerta a opensource.

La nueva versión de DD-WRT (v24) es un proyecto completamente nuevo. DD-WRT ofrece muchas características avanzadas que no se encuentran en el firmware de estos dispositivos OEM, o incluso el firmware disponible para la compra en Sveasoft. Está también libre de la activación de producto o del seguimiento del firmware por Sveasoft.

- ***OpenWrt:*** En lugar de crear un firmware único y estático, OpenWrt proporciona un sistema de archivos totalmente modificable con la gestión de paquetes. Esto le libera de la selección de aplicaciones y la configuración proporcionada por el vendedor y le permite personalizar el dispositivo incorporando aplicaciones a través del uso de paquetes. Para los desarrolladores, OpenWrt es el marco para construir una aplicación sin tener que construir un firmware completo a través de ella. Para los usuarios, esto significa una capacidad de personalización completa, para utilizar un dispositivo embebido en formas que nunca imagino.


**Imagen 3.3: Logo OpenWrt**

Algunas de las características de OpenWrt:

- ✓ *Libre y de código abierto:* El proyecto es gratuito y de código abierto (GPL). El proyecto tiene la meta de estar alojado en un sitio accesible a todo el mundo, con el código fuente al completo para que se pueda desarrollar fácilmente.

- ✓ *Fácil y de libre acceso:* El proyecto está abierto a nuevas contribuciones, cualquier persona puede ser capaz de contribuir con OpenWrt. Los desarrolladores conceden acceso a cualquier persona que lo solicite, ya que piensan que las personas son responsables cuando se les da responsabilidad.

- ✓ *Impulsado por la comunidad:* No se trata de un proyecto en el que los desarrolladores ofrecen algo a los usuarios, se trata de que todos trabajen juntos para lograr una meta común.

OpenWrt ha demostrado durante mucho tiempo que es la mejor solución de firmware de su clase. Es muy superior a otras soluciones en rendimiento, estabilidad, extensibilidad, robustez y diseño. La meta de los desarrolladores es continuar y ampliar OpenWrt para garantizar que es el sistema favorito para el desarrollo y la innovación.

### 3.3. Solución elegida.

La solución que se va a desarrollar en este proyecto fin de carrera será ejecutar el protocolo de comunicaciones W2LAN.

Para desarrollar el proyecto necesitaremos seleccionar un router que nos permita instalar una distribución de Linux, además necesitaremos las herramientas de desarrollo del firmware seleccionado, como compilar una imagen del sistema operativo con los paquetes que necesitemos, un SDK para poder compilar y crear el paquete de W2LAN y que el firmware nos permita la instalación de paquetes.

El firmware que utilizaremos será OpenWrt, que está basado en el kernel de Linux, ofrece herramientas de desarrollo y tiene una gran documentación en la web del proyecto. En concreto utilizaremos la versión de OpenWrt, llamada "kamikaze 8.09" (Septiembre 2008) que es la versión más actual en el momento de comenzar este proyecto, la versión de kernel que se utilizara es la 2.4 ya que el router dispone de un chip broadcom y la empresa desarrolladora no ha liberado los driver necesarios para el kernel 2.6.

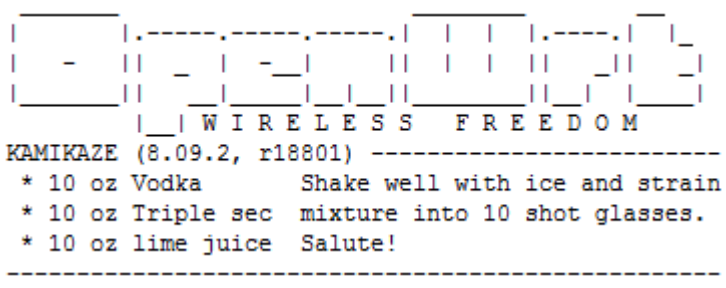



**Imagen 3.3: Encabezado del firmware OpenWrt**

Una vez seleccionado el router (WRT54GL) y el firmware (OpenWrt 8.09 kamikaze), necesitaremos familiarizarnos con las herramientas de desarrollo que nos proporciona este firmware (SDK e Image Builder) y las herramientas para la configuración del router a nivel de firmware.

Una vez compilado, creado e instalado el paquete de W2LAN en el router con la distribución de OpenWrt, el router será capaz de integrarse en una red que funcione bajo este protocolo de comunicaciones de nivel 2.

# 4. Herramientas necesarias.

## 4.1. W2Lan.

### *4.1.1. Introducción.*

El protocolo W2LAN (Wireless to LAN) es un protocolo de nivel de enlace que transforma desde el punto de vista de capas superiores una red 802.11x MANET en una red LAN Ethernet, manteniendo compatibilidad con otros protocolos ya bien establecidos y probados. Así, usando W2LAN cualquier par de terminales inalámbricos que pertenezcan a la misma LAN se verán entre ellos - visibilidad total-, siempre que al menos exista una ruta entre ellos (característica mínima necesaria en cualquier red).

### *4.1.2. Servicio y suposiciones.*

El servicio que el protocolo W2LAN ofrece es dotar de visibilidad total a los nodos pertenecientes a una red Ad-Hoc determinada. Cuando un nodo tiene una trama Ethernet preparada para transmitir, en lugar de transmitir la trama directamente, ésta es entregada a la capa W2LAN, que de inmediato iniciará una nueva conversación W2LAN. De manera complementaria, cuando un nodo ha recibido una conversación completa, su capa W2LAN entregará la trama Ethernet asociada a dicha conversación igual que lo haría un dispositivo Ethernet normal.

El protocolo W2LAN se basa en 2 suposiciones. La primera es que opera sobre dispositivos wireless Ethernet en modo Ad-Hoc. Esto significa que los nodos deben de tener al menos un dispositivo wireless configurado correctamente, tal como se haría en una red Ad-Hoc sin W2LAN. La segunda suposición es que debe existir al menos una ruta posible entre cualquier par de nodos. El protocolo sigue funcionando en redes fragmentadas, tratándolas como redes aisladas. Ahora bien, se sobreentiende que no es el caso deseado, ya que el hecho de usar W2LAN lleva implícito que se está intentando solucionar un problema de visibilidad parcial.

### *4.1.3. Vocabulario y formato de las tramas.*

El protocolo W2LAN se basa en el concepto de transacción W2LAN (subconjunto de tramas pertenecientes a una misma conversación atendidas por un nodo) y el de conversación (conjunto de tramas en la red que comparten el mismo identificador de conversación). Para describir el vocabulario es suficiente con desglosar las transacciones. Una transacción W2LAN consiste en una tripleta {anuncio, solicitud, datos} que transportará tramas Ethernet a los distintos nodos vecinos interesados.

El vocabulario del protocolo W2LAN necesario para construir transacciones W2LAN será:

- ***Anuncio:*** El mensaje anuncio es utilizado cuando un nodo tiene una nueva trama a transmitir, ya sea proveniente de capas superiores o proveniente de una conversación acabada de recibir que debe ser retransmitida. Este mensaje lleva asociado la dirección MAC origen del nodo, un identificador (ID) de conversación y la cabecera de la trama Ethernet que se está distribuyendo.

- ***Solicitud:*** El mensaje solicitud se utiliza la primera vez que un nodo recibe un anuncio de una conversación nueva. Este mensaje lleva asociado la dirección MAC destino del nodo (o de manera equivalente, la dirección MAC origen del nodo que mandó el correspondiente anuncio) y el identificador de conversación correspondiente.

- ***Datos:*** El mensaje datos se utiliza si un nodo que ha anunciado previamente una conversación recibe alguna solicitud de la conversación anunciada. Este mensaje lleva asociado el correspondiente identificador de conversación y el campo de datos de la trama Ethernet que se está distribuyendo. El formato de cada mensaje se ha diseñado para que sea compatible con Ethernet, teniendo en cuenta en todo momento la simplicidad. Así, cada mensaje encaja en una trama Ethernet, evitando fragmentar los mensajes.

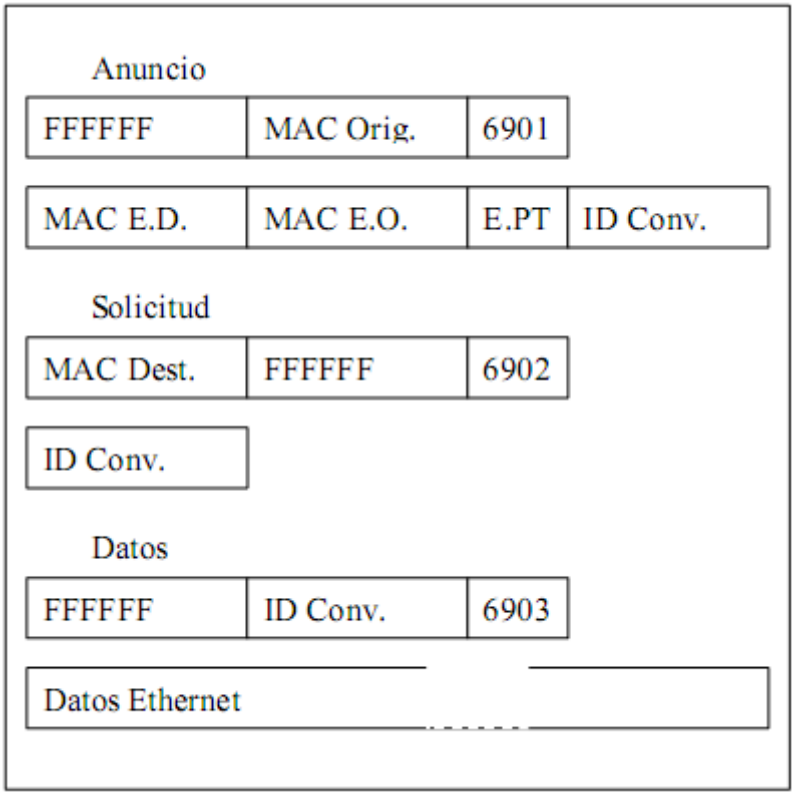



**Imagen 4.1: Tramas (cabecera/datos) que utiliza W2LAN**

## 4.2. OpenWrt.

### *4.2.1. Introducción.*

Con el lanzamiento del código fuente de Linux para las series de routers Linksys WRT54G/GS, vino un número de firmwares modificados para extender la funcionalidad en varias direcciones. Cada firmware era el 99% del código fuente proporcionado por Linksys y el otro 1% de funcionalidades añadidas, y cada firmware intentaba atender a un cierto segmento del mercado con la funcionalidad que el firmware proveía. Los inconvenientes:

- A menudo era difícil encontrar un firmware con la combinación de la funcionalidad deseada.

- Todos los firmwares estaban basados en el código fuente original de Linksys, que estaba lejos del actual desarrollo de GNU/Linux.

OpenWrt toma un camino distinto, en lugar de iniciarse con el código fuente de Linksys, el desarrollo comenzó desde cero. Cada parte del software se añadió para volver a ofrecer la funcionalidad de nuevo que existía en los distintos firmwares, usando las más recientes versiones disponibles. Lo que hace a OpenWrt realmente único es el hecho del empleo de un sistema de archivos con posibilidad de escritura, por lo que el firmware ya no es sólo un firmware con una compilación estática de software, sino que es posible instalar dinámicamente ajustándose a las necesidades para cualquier tipo de uso particular. En poco tiempo, el dispositivo se convirtió en una mini-distribución de Linux con OpenWrt actuando como la distribución completa con casi todos los comandos tradicionales de Linux y un sistema de gestión de paquetes para cargar software y características adicionales.

### *4.2.2. Por qué se debe usar OpenWrt.*

Porque GNU/Linux proporciona la capacidad de hacer lo que se necesita con un hardware barato y evitando usar software propietario e inflexible. OpenWrt no es para gente de pocos conocimientos, pero si se tiene la necesidad de usar un barebone GNU/Linux y se está preparado para hacer algún trabajo, OpenWrt es el firmware más rápido basado en Linux para una gran cantidad de routers con Ethernet y wireless. En este momento, la distribución contiene más de 100 paquetes de software. Además, la comunidad de OpenWrt proporciona muchos más paquetes añadidos. Para los desarrolladores, el proyecto proporciona un sistema de construcción con el que es posible crear un firmware modificado desde el código fuente. La forma de añadir nuevos paquetes es sencilla con el uso del SDK.

### *4.2.3. Historia de las versiones de OpenWrt.*

El proyecto comenzó en Enero de 2004. La primera versión de OpenWrt estaba basada en el código fuente GPL de Linksys para WRT54G y construido desde el proyecto uclibc. Esta versión era conocida como una versión de OpenWrt estable y tuvo un extenso uso.

Hay aún muchas aplicaciones de OpenWrt, como el Freifunk-Firmware o el Sip@Home, que están basados en esta versión.

En los comienzos de 2005 se unieron nuevos desarrolladores al equipo. Tras algunos meses de desarrollo cerrado, el equipo decidió publicar la primera versión experimental de OpenWrt. Las versiones experimentales usan un sistema de construcción robusto y personalizado basado en buildroot2 del proyecto uclibc. OpenWrt usa el código fuente oficial de GNU/Linux y sólo añade parches para el sistema de chip y controladores para las interfaces de red. El equipo de desarrollo intenta reimplementar gran cantidad del código propietario dentro de los códigos fuente GPL de distintas empresas. Hay utilidades libres para escribir imágenes nuevas de firmware directamente dentro de la memoria flash (mtd), para configurar los chips de red wireless (wlcompat/wificonf) y programar el switch VLAN a través del sistema de archivos proc. El nombre en código del primer lanzamiento de OpenWrt fue White Russian, un cóctel popular.

En septiembre de 2007, se empezó a utilizar el nombre en clave kamikaze, lo más reseñable de esta versión es el abandono de la NVRAM para poder extender es uso de OpenWrt a la mayor parte de dispositivos, ya que no todos los router disponen de esta memoria. Existen dos versiones de kamikaze la versión 7.09 (Septiembre 2007) y 8.09 (Septiembre de 2008).

Actualmente, la última línea de desarrollo se denomina con el nombre en código de backfire y la versión de este es 10.03 (Marzo de 2010).

#### *4.2.4. Obtención del firmware.*

Hay principalmente dos formas de obtener el firmware de OpenWrt:

- Podemos obtener el firmware desde la web de OpenWrt y descargarte desde esta el archivo del firmware para tu dispositivo. (http://downloads.openwrt.org/kamikaze/8.09.2/)

- Podemos obtener el firmware del dispositivo es mediante el repositorio de subversión (SVN) y crearnos nuestra propia imagen del firmware con los paquetes que nosotros seleccionaremos. Esta forma de obtener el firmware se explicará posteriormente en el apartado **4.3.2** de este proyecto de final de carrera.

#### *4.2.5. Tipos de imagen del firmware: TRX y BIN.*

Los archivos del firmware están distribuidos por ambos tipos de archivos "trx" o "bin". El archivo bin no es más que un archivo trx con una información adicional añadida para hacerlo compatible con las utilidades de actualización proporcionadas por las versiones de firmware originales. Se debe de usar nada más que este archivo cuando no es posible usar archivos trx de forma directa.

Versiones disponibles para algunos routers:

- openwrt-brcm-2.4-<type>.trx

Este es el firmware en formato RAW, exactamente como será escrito en la memoria flash. Este formato se usa al actualizar desde dentro de OpenWrt o durante la instalación inicial en cualquiera de los siguientes routers:

    - Asus WL500g
    - Asus WL500gx
    - Buffalo Airstation WBR2-G54S
    - WRT54G v5 - v6 (sólo, sin soporte oficial)
    - WRT54GS v5 - v6 (sólo, sin soporte oficial)

- openwrt-wrt54g-<type>.bin; openwrt-wrt54gs-<type>.bin; openwrt-wrt54gs_v4-<type>.bin; openwrt-wrtsl54gs-<type>.bin

Este es exactamente lo mismo que el archivo trx anterior, con una excepción: se ha añadido una pequeña cabecera al comienzo del archivo, marcándolo como una actualización válida para modelos de Linksys. Los modelos soportados son:

- openwrt-wrt54g-<type>.bin
    - Linksys WRT54G (v1.0, v1.1, v2.0, v2.2, v3.0, v4.0)
    - Linksys WRT54GL
- openwrt-wrt54gs-<type>.bin
    - Linksys WRT54GS (v1.0, v1.1, v2.0, v3.0)
- openety-wrt54g3g-<type>.bin
    - Linksys WRT54G3G
- openwrt-wrt54gs_v4-<type>.bin
    - Linksys WRT54GS (v4.0)
- openwrt-wrtsl54gs-<type>.bin
    - Linksys WRTSL54GS
- openwrt-wa840g-<type>.bin; openwrt-we800g-<type>.bin; openwrt-wr850g-<type>.bin

Este también es un archivo  trx, pero con una cabecera de Motorola añadida al comienzo del archivo, haciéndolo un archivo de firmware válido para un dispositivo Motorola.

Tras descargar la imagen del firmware, hay que cerciorarse de que el archivo no está corrupto. Se puede verificar mediante la comparación de la suma MD5 de la imagen descargada con la suma MD5 listada en los archivos de sumas de MD5 disponibles en el directorio de descargas. Para plataformas win32, mediante el uso de md5sums.exe y para plataformas GNU/Linux, mediante el comando md5sum.

### *4.2.6. Instalando OpenWrt.*

Hay muchas formas de instalar el firmware, se explicará cada método a continuación. Se puede usar cualquier método, al final el resultado será el mismo. Tras instalar el firmware, el dispositivo se reiniciará automáticamente con el nuevo firmware. Si no se está contento con OpenWrt, siempre se puede reinstalar el firmware original.

- ***Por medio de la pagina web del firmware original***: Este es el método más sencillo. Abriendo el navegador web y usando la página de actualización de firmware del dispositivo para instalar el firmware de OpenWrt.

- ***A través de TFTP:*** Si se hace con mucha prudencia o se está intentando reinstalar tras una actualización fallida, se puede usar tftp para instalar el firmware. Este método está explicado a continuación. Los pasos para instalar el firmware por medio de un cliente tftp es prácticamente similar en cualquier sistema operativo. En cualquiera de los tres sistemas, se tendrá que configurar el ordenador con la IP estática 192.168.1.10 y con la máscara 255.255.255.0.

Para usar el cliente TFTP se verá cómo hacerlo en cada sistema:

a) Instalación desde tftp en Linux/BSD:

El router tiene que estar apagado, es decir, el cable de alimentación debe estar desconectado. Se accederá a la ruta en donde se encuentra el firmware OpenWrt descargado desde la consola de comandos, y se ejecutarán los siguientes comandos: tftp 192.168.1.1

Binary

rexmt 1

timeout 60

trace

put openwrt-xxx-x.x-xxx.bin

La opción rexmt 1 se utiliza para que el cliente reintente constantemente enviar el archivo a la dirección dada. Una vez introducidos los comandos, se conectará a la corriente el router y cuando el cargador de arranque del router comience a esperar el envío del firmware, automáticamente se le enviará. El proceso tarda un tiempo y en ningún momento se debe apagar el router. Cuando el firmware esté actualizado el router se reiniciará de nuevo.

b) Instalación desde tftp en Mac OS X.

Para Mac OS se recomienda usar el cliente MacTFTP en puesto del cliente tftp que viene integrado en el sistema, debido a ciertos fallos que han sido reportados por algunos usuarios.

1) Descargar MacTFTP:

   http://www.mactechnologies.com/pages/downld.html

2) Elegir la opción de envío.

3) Introducir la dirección del router: 192.168.1.1

4) Seleccionar el archivo que se va a enviar: openwrt-xxx-x.x-xxx.bin

5) Y finalmente pulsar en iniciar y conectar el router a la corriente.

c) Instalación desde tftp en Windows 2000/XP/Vista/7.

Habrá que desactivar cualquier cortafuegos que se tenga instalado para poder enviar el firmware a través del cliente tftp.

Los pasos para instalar son los siguientes:

1) Abrir dos consolas de comandos de Windows.

2) En la primera ejecutar: ping -t -w 10 192.168.1.1

3) El ping estará intentando conectar con el router cada 10ms constantemente.

4) En la segunda acceder a la ruta donde se encuentra el firmware OpenWrt.

5) Y ahora escribir: tftp -i 192.168.1.1 PUT openwrt-xxx-x.x-xxx.bin pero no presionar la tecla Intro (Enter).

6) Conectar el router y se mostrará en la primera ventana que aparece "Error de Hardware".

7) Tan pronto como vuelve a funcionar el ping, se ejecuta el comando de tftp de la segunda ventana y se enviará el archivo al router.

- ***A través de CFE:*** Si ya se tiene un cable de conexión por puerto serie, se puede utilizar para instalar el firmware a través de él.

- ***A través de JTAG***: No se recomienda instalar la imagen del kernel por medio de JTAG, ya que tardará alrededor de 2 horas, pero es posible instalarlo.

- ***A través de la línea de comandos de OpenWrt:*** Actualizar OpenWrt sobrescribirá el sistema de archivos, borrando todos los datos y aplicaciones anteriores. Es extremadamente necesario que se haga una copia de seguridad con cualquier cambio que se haya hecho en el sistema.

  mtd -r write firmware.trx linux

Para modelos con poca capacidad de memoria, se puede instalar el firmware directamente desde un sitio web.

wget http://www.unsitioweb.com/firmware.trx -O - | mtd -e Linux -r write – Linux

#### *4.2.7. Usando OpenWrt por primera vez.*

1) Arrancando

En routers con un LED para DMZ, OpenWrt usará este LED como señal de arranque, encendiendo LED mientras arranca y apagándolo una vez que se ha completado el arranque.

Una vez que ha arrancado, se deberá poder realizar un telnet para acceder al router, usando la dirección IP 192.168.1.1.

2) Asignando la contraseña.

En este punto, es extremadamente importante que se asigne una contraseña. Dependiendo de cada imagen de firmware que se tenga instalada, se puede fijar o bien mediante la página web de OpenWrt (Webif) o mediante telnet usando el comando passwd. Tras asignar la contraseña, cualquier intento de telnet tendrá como resultado el mensaje "Login failed", porque se habrá deshabilitado telnet en favor de SSH.

*¿Por qué no hay una contraseña por defecto?*

Las personas son muy perezosas. No se quiere dar a la gente un sentimiento falso de seguridad creando una contraseña que cualquiera pueda conocer. Se quiere estar seguro de que se sabe que no hay seguridad para incitarle al usuario que asigne una contraseña.

*¿Qué pasa si no se puede acceder por telnet la primera vez que se arranca?*

Esto puede ser un buen problema en relación con los ajustes del firewall en Linux o Windows. Si hay algún firewall, se debe desactivar. Sin embargo, una vez que OpenWrt está instalado y se reinicia por primera vez, telnet no funcionará nunca más (por razones de seguridad). Se debe usar SSH como alternativa. Hay muchos buenos clientes para SSH (Tunnelier, Putty, etc).

*¿Qué pasa si no se puede acceder por SSH tras asignar una contraseña?*

Se debe intentar tras esperar un minuto o dos. En el primer arranque OpenWrt estará ocupado ajustando el sistema de archivos y generando las claves de cifrado de SSH; el servidor SSH no se iniciará hasta que se hayan generado las claves de cifrado.

*¿Por qué se rechaza la contraseña o muestra advertencias de SSH tras actualizar?*

Actualizar OpenWrt por completo, reemplaza el sistema de archivos. Esto significa que la contraseña anterior y las claves de cifrado de SSH se eliminarán y se tendrá que asignar una contraseña de nuevo.

3) Gestión de paquetes.

La utilidad opkg es un gestor de paquetes ligero usado para descargar e instalar paquetes de OpenWrt a través de internet. (A los usuarios de GNU/Linux les parecerá similar al comando apt-get). El firmware está diseñado por sí solo para ocupar el menor espacio posible mientras que provee de una razonable interfaz de comandos amigable o una interfaz web de administración. Sin paquetes instalados, OpenWrt configurará las interfaces de red, ajustará un firewall NAT básico, un servidor seguro de shell, un servidor DNS y un servidor DHCP.

Se pueden encontrar más opciones a través del comando opkg --help

El archivo de configuración de opkg es /etc/opkg.conf

*¿Cómo instalar paquetes?*

Para instalar un paquete que esté presente en un repositorio (por lo general este es el caso):

opkg install wireless-tools

Para instalar un paquete desde una dirección URL o archivo descargado:

opkg install http://downloads.openwrt.org/kamikaze/8.09.1/brcm-2.4/packages/openssh-sftp-server_5.0p1-1_mipsel.ipk

En ambos casos, opkg tratará de resolver las dependencias con los paquetes en los repositorios. Si no se puede, se informará de un error, y no se instalaran ninguno de los paquetes.

*Como listar los paquetes instalados.*

opkg list_installed

*Como borrar paquetes.*

opkg remove

*Como actualizar el sistema.*

Para actualizar el sistema se debe utilizar: opkg upgrade

No se recomienda actualizar de esta forma OpenWrt, ya que los paquetes descargados pueden llegar a llenar la memoria interna del dispositivo, por lo que para actualizar el sistema es aconsejable realizar un flasheo con una imagen actualizada.

### *4.2.8. Configuración de OpenWrt.*

#### *4.2.8.1. El sistema UCI.*

La abreviatura "uci" significa "Configuración de la interfaz unificada". Es el sucesor del comando nvram de configuración basado en la serie White Russian de OpenWrt.

#### *4.2.8.2. Los principios comunes.*

En OpenWrt la configuración central está dividida en varios archivos ubicados en el directorio /etc/config/. Cada archivo corresponde a la parte del sistema que configura. La edición de los archivos de configuración se puede realizar con un editor de texto, la sentencia UCI o mediante varios APIs de programación (como Shell, Lua y C).

#### *4.2.8.3. Archivos de configuración.*

| Archivo | Descripción |
|---|---|
| /etc/config/dhcp | Configuración dnsmasq y ajustes DHCP |
| /etc/config/dropbear | Opciones del servidor SSH |
| /etc/config/firewall | Configuración del firewall |
| /etc/config/fstab | Puntos de montaje y de intercambio |
| /etc/config/httpd | Opciones del servidor Web |
| /etc/config/network | Configuración del Switch, de la interfaz y de la tabla de encaminamiento |
| /etc/config/ntpclient | Obtener la hora correcta |
| /etc/config/samba | configuraciones para el archivo M$ y servicios de impresión |
| /etc/config/system | Configuración del sistema |
| /etc/config/timeserver | Lista de servidores de tiempo para *rdate* |
| /etc/config/wireless | Configuración inalámbrica y la definición de red wifi |
| /etc/config/wol | Ajustes para Wake on LAN (*WOL*). |

#### *4.2.8.4. Sintaxis de los archivos de configuración.*

La configuración uci archivos suelen consistir en una o más instrucciones de configuración, llamados secciones con las declaraciones de uno o más opción de definir los valores reales.

A continuación se muestra un ejemplo de una configuración simple de archivos:

```
package 'example'

config 'example' 'test'

        option  'string'        'some value'

        option  'boolean'       '1'

        list    'collection'    'first item'

        list    'collection'    'second item'
```

- La sentencia config 'example' 'test' define el inicio de una sección de tipo 'example' y de nombre 'test'. También puede crear una sección anónima que solo posea el tipo, pero que carezca del identificador de nombre. El tipo es importante para los programas ya que les indica como tienen que tratar las opciones.

- Las líneas option 'string' 'some value' y option 'boolean' '1' líneas definen valores simples dentro de la sección. Tenga en cuenta que no hay diferencias sintácticas entre booleanos y opciones de texto. Las opciones booleanas pueden tener uno de los siguientes valores '0 ',' no ',' off 'o' false 'para especificar un valor falso o '1', 'yes', 'on' o 'true' para especificar un valor verdadero.

- Las líneas que comiencen con la sentencia list definen una opción con varios valores. Todos los elementos de la lista serán guardados en una variable, en nuestro caso 'collection', en el mismo orden que en el archivo de configuración.

En general, no es necesario encerrar los identificadores o los valores entre comillas. Las comillas sólo son necesarias si el valor contiene espacios o tabuladores. También es legal el uso doble en vez de comillas simples al escribir opciones de configuración.

Todos estos ejemplos tienen una sintaxis correcta para UCI:

- option example value
- option 'example' value
- option example " value"
- option "example" 'value'
- option 'example' "value"

Es importante saber que los identificadores de configuración de la UCI y los nombres de archivo sólo pueden contener los caracteres az, 0-9 y _. Los valores de opción pueden contener cualquier carácter.

#### *4.2.8.5. Configuración de la red.*

El modelo WRT54G consta de un switch Ethernet, un punto de acceso Wireless y un chip del router que los conecta agrupados. Su diagrama es el siguiente:

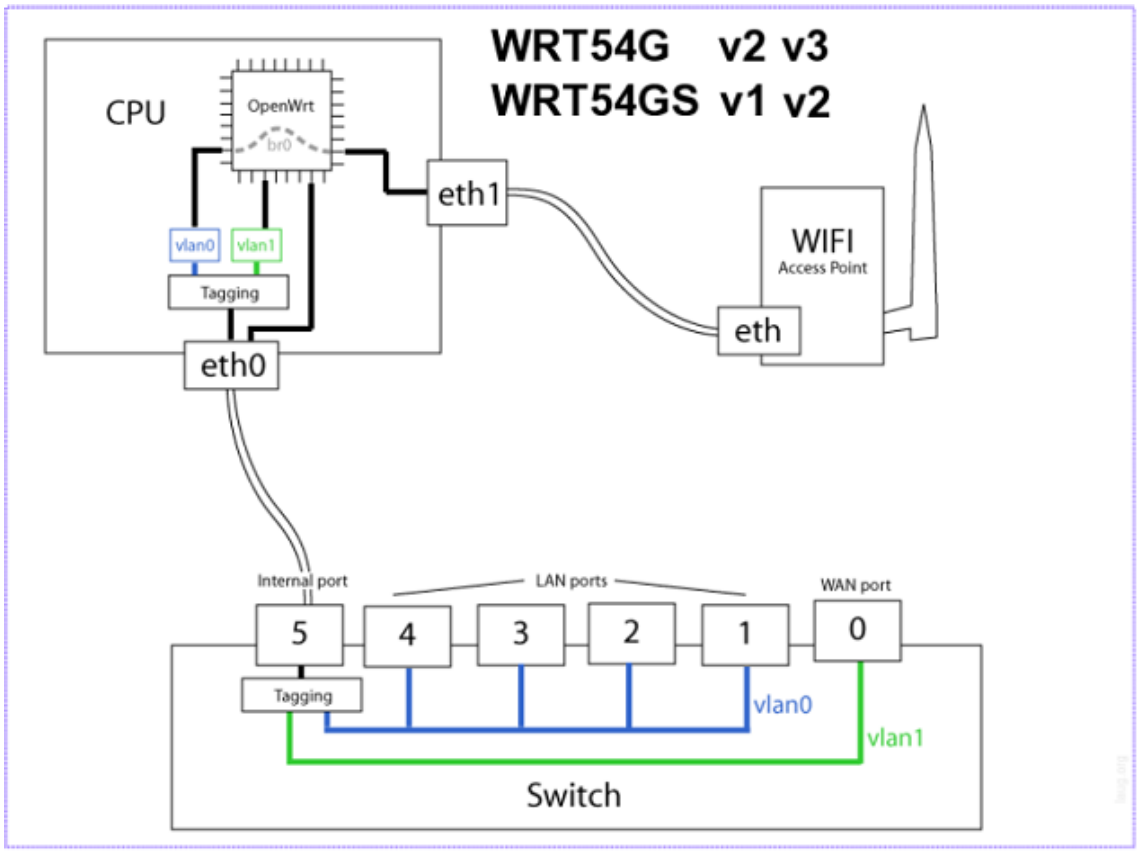



**Imagen 4.2: Diagrama de configuración de la red.**

Los nombres de las interfaces de red dependerán del hardware en el que se está ejecutando OpenWrt.

Para el modelo WRT54GL las interfaces son:

LAN → vlan0

WAN → vlan1

WIFI → eth1

***NOTA:*** LAN y WIFI están puenteados juntos en la interfaz br0 por defecto, en algunos dispositivos WAN puede ser eth1 y LAN eth0.

#### *4.2.8.6. Configuración del archivo network.*

La configuración de la red se encuentra en el archivo /etc/config/ network. Esta configuración se encarga de definir las VLAN del switch, la configuración de las interfaces y las tablas de encaminamiento.

- ***Secciones***

A continuación se muestra una visión general de los tipos que se pueden definir en la configuración de la red. Una configuración de red mínima de un router por lo general consta de al menos dos *interfaces* (LAN y WAN) y un switch (si este está presente en el router).

- ***Switch***

La sección del switch es responsable de separar el switch en varias VLAN que aparecen como interfaces independientes en el sistema, aunque comparten el mismo hardware. No todos los dispositivos con OpenWrt tienen un switch configurable, por lo que esta sección puede no estar presente en algunas plataformas.

Por el momento hay dos formas posibles de configuración del switch, mediante la API /proc/switch/ y mediante swconfig, basada en la infraestructura del switch.

- ***/proc/switch***

Esta variante en la configuración, solo está disponible en los dispositivos con chip Broadcom, como el WRT54GL.

La configuración típica es:

```
config 'switch' 'eth0'

    option 'vlan0' '0 1 2 3 5*'

    option 'vlan1' '4 5'
```

El identificador eth0 especifica a que sección del switch pertenece. Las VLAN se definen mediante ‘vlan#’ donde # representa el numero de VLAN.

- ***Interfaces***

La sección de la interfaz, juega un papel importante en la configuración de las redes ya que en ella se declara la dirección IP, la máscara de red, las rutas, las reglas del firewall y el nombre de la interfaz.

Una declaración de interfaz mínima se compone de las siguientes líneas:

```
config 'interface' 'wan'

    option 'proto' 'dhcp'

    option 'ifname' 'eth0.1'
```

- Wan es el nombre de la interfaz lógica.

- dhcp especifica el protocolo a seguir de la interfaz (DHCP en este ejemplo)
- eth0.1 es la interfaz física asociada a esta configuración.

Los protocolos que puede seguir la interfaz son los siguientes:

| Protocolo | Descripción |
|---|---|
| static | Configuración con una dirección IP y mascara fijas |
| dhcp | Dirección y máscara de red se asignan por DHCP |
| ppp | protocolo PPP - conexiones de módem telefónico |
| pppoe | PPP sobre Ethernet - Conexión de banda ancha DSL |
| pppoa | PPP sobre ATM - conexión mediante un módem DSL incorporado |
| 3g | CDMA, UMTS o GPRS utilizando un módem AT (modem 3G) |
| pptp | Conexión a través de PPTP VPN |
| none | Sin especificar el protocolo |

Dependiendo del protocolo de interfaz seleccionado, es posible que se necesiten otras opciones para completar la configuración.

- **Las opciones válidas para todos los tipos de protocolo**

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| ifname | nombre de la interfaz | sí | *(Ninguno)* | nombre de la interfaz física, o si el bridge esta activo la lista de interfaces |
| type | string | no | *(Ninguno)* | Si se crea un bridge, contiene las interfaces de “ifname” |
| stp | booleano | no | 0 | Sólo es válido para bridge, activa Spanning Tree Protocol |
| macaddr | dirección MAC | no | *(Ninguno)* | Cambia la MAC de la interfaz |
| mtu | número | no | *(Ninguno)* | Cambia el MTU por defecto en esta interfaz |
| auto | booleano | no | 0 si no se pone protocolo, de lo contrario 1 | Especifica si se tiene que activar la interfaz en el arranque. |

- **Protocolo "static"**

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| ipaddr | dirección IP | si | *(Ninguno)* | Dirección IP |
| netmask | máscara de red | si | *(Ninguno)* | Máscara de red |
| Gateway | dirección IP | no | *(Ninguno)* | Puerta de enlace predeterminada |
| bcast | dirección IP | no | *(Ninguno)* | dirección de difusión (generado automáticamente si no se establece) |
| dns | lista de direcciones IP | no | *(Ninguno)* | DNS Server (s) |

- **Protocolo "dhcp"**

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| gateway | string | no | *(Ninguno)* | Reemplaza la asignación de dhcp si se pone en 0.0.0.0 |
| dns | lista de direcciones IP | no | *(Ninguno)* | Reemplaza la asignación DHCP por la DNS del servidor |

- **Alias**

Esta sección se utiliza para definir las direcciones IP de las interfaces. También permiten combinaciones de direcciones dhcp en la interfaz y direcciones estáticas en el alias. Cada interfaz puede tener múltiples alias.

Una declaración de alias mínima consta de las siguientes líneas:

```
config 'alias'
        option 'interface' 'lan'
        option 'proto' 'static'
        option 'ipaddr' '10.0.0.1'
        option 'netmask' '255.255.255.0'
```

- LAN es el nombre de la interfaz lógica del padre.
- static es el protocolo de la interfaz.
- 10.0.0.1 especifica la dirección IP.
- 255.255.255.0 especifica la máscara de red.

Hasta el momento solo se permite la configuración estática del alias. Las opciones para alias se muestran a continuación:

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| interface | string | sí | *(Ninguno)* | Especifica el nombre de la interfaz lógica a la que pertenece, esta interfaz lógica debe configurarse anteriormente. |
| proto | string | sí | *(Ninguno)* | Especifica el protocolo de la interfaz |
| ipaddr | dirección IP | sí | *(Ninguno)* | Dirección IP |
| netmask | máscara de red | sí | *(Ninguno)* | Máscara de red |
| gateway | dirección IP | no | *(Ninguno)* | Puerta de enlace predeterminada |
| bcast | dirección IP | No | *(Ninguno)* | dirección de broadcast (generado automáticamente si no se establece) |
| dns | lista de direcciones IP | No | *(Ninguno)* | Servidor DNS |

- **Tabla de encaminamiento.**

Podemos definir la tabla de encaminamiento en la interfaz específica con la sección route.

Una configuración básica:

```
config 'route'
        option 'interface' 'lan'
        option 'target' '172.16.123.0'
        option 'netmask' '255.255.255.0'
```

- LAN es el nombre de la interfaz lógica del padre.
- 172.16.123.0 es la dirección de la red de la ruta.
- 255.255.255.0 especifica la máscara de red.

Las opciones validas para las tablas de encaminamiento son:

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| interface | string | sí | *(Ninguno)* | Especifica el *nombre de la interfaz lógica* a la que pertenece esta ruta, debe hacer referencia a una de las interfaces definidas anteriormente. |
| target | dirección IP | sí | *(Ninguno)* | direcciones de red |
| netmask | máscara de red | no | *(Ninguno)* | Máscara de red. Si se omite, 255.255.255.255 se asume que hace referencia a una dirección de host. |
| gateway | dirección IP | no | *(Ninguno)* | Puerta de enlace. Si se omite se toma la de los padres y si se establece a 0.0.0.0 no hay puerta de enlace. |
| metric | número | No | 0 | Especifica la métrica a utilizar |

- **Ejemplos**

A continuación se muestran algunos ejemplos de configuraciones distintas a las que vienen por defecto.

- **Bridge sin IP**

```
config 'interface' 'example'
        option 'type'    'bridge'
        option 'proto'   'none'
        option 'ifname'  'eth0 eth1'
        option 'auto'    '1'
```

- **DHCP sin puerta de enlace predeterminada**

```
config 'interface' 'example'
        option 'proto'   'dhcp'
        option 'ifname'  'eth0'
        option 'gateway' '0.0.0.0'
```

- **Configuración de IP estática y puerta de enlace predeterminada con métricas distintas de cero**

```
config 'interface' 'example'
        option 'proto'     'static'
        option 'ifname'    'eth0'
        option 'ipaddr'    '192.168.1.200'
        option 'netmask'   '255.255.255.0'
        option 'dns'       '192.168.1.1'
config 'route'
        option 'interface' 'example'
        option 'target'    '0.0.0.0'
        option 'netmask'   '0.0.0.0'
        option 'gateway'   '192.168.1.1'
        option 'metric'    '100'
```

#### *4.2.8.7. Configuración del archivo wireless.*

La configuración inalámbrica se encuentra en /etc/config/wireless. Este archivo se encargará de definir los dispositivos y las redes wireless.

Al arrancar por primera vez el sistema, este rellenará el archivo de configuración con algunos valores básicos el cual se puede editar para adaptar la configuración.

- **Secciones**

El archivo de configuración inalámbrica contiene al menos un dispositivo wifi que especifica las propiedades generales de radio, tales como el canal, tipo de controlador y potencia de transmisión y una interfaz wireless necesaria para definir una red inalámbrica.

- **Dispositivos Wifi**

Dispositivos wifi hace referencia a dispositivos físicos que estén presente en el sistema. Las opciones presentes en esta sección describen propiedades comunes a todas las redes inalámbricas con dispositivos radio como canal o selección de antena.

En la mayoría de los casos sólo hay un adaptador inalámbrico disponible en el dispositivo, así que sólo se define una de estas secciones, sin embargo en los dispositivos con múltiples adaptadores wireless, se creara un apartado por cada una de ellas.

Una declaración mínima del archivo de configuración de un dispositivo wifi se muestra a continuación.

```
config 'wifi-device' 'wl0'

        option 'type'    'broadcom'

        option 'channel' '6'
```

- wl0 es el identificador del adaptador inalámbrico.
- Broadcom especifica el conjunto de chips y el tipo de controlador.
- 6 es el canal en el que funciona el dispositivo.

Las opciones posibles de la configuración de los dispositivos inalámbricos se muestran a continuación.

- **Opciones Comunes**

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| type | String | sí | autodetectado | El tipo se determina en el arranque inicial (por lo general no es necesario cambiarlo). Los valores utilizados son: broadcom para brcm-2.4, atheros para madwifi o mac80211 para b43, ath5k y ath9k. |
| disabled | boolean | no | 1 | 1 desactiva el adaptador inalámbrico y con el valor 0 activa el adaptador. |
| channel | integer o "auto" | sí | auto | Especifica el canal inalámbrico. |
| hwmode | String | no | Driver por defecto | Selecciona el protocolo inalámbrico a utilizar: 11a, 11b, 11g, 11n o automático |
| txpower | Integer | no | Driver por defecto | Potencia en dBm |
| diversity | Boolean | no | 1 | Activa o desactiva la selección automática de la antena por el driver |
| rxantenna | Integer | no | Driver por defecto | Especifica la antena de recepción: 1 para la primera antena, 2 para la segunda o 0 automática. Esta opción no tiene efecto si diversity está habilitado |

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| txantenna | Integer | no | Driver por defecto | Especifica la antena de transmisión. (Resto ídem rxantenna) |
| macfilter | String | no | desactivado | Especifica el filtrado de MAC, se puede tratar como lista blanca o lista negra. |
| maclist | lista de direcciones MAC | no | Ninguno | Lista de direcciones MAC para poner en macfilter. |
| país | String (2 letras) | no | Driver por defecto | Especifica el código del país, afecta a los canales disponibles y potencias de transmisión. |

- **Opciones Broadcom**

Las siguientes opciones son usadas únicamente por el controlador propietario de Broadcom.

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| frameburst | boolean | no | 0 | Habilita la trama broadcom si es soportada. |
| maxassoc | entero | no | Driver por defecto | Limita el número máximo de clientes asociados |
| slottime | entero | no | Driver por defecto | Ranura de tiempo en milisegundos |

- **Redes Wifi**

Una configuración inalámbrica completa contiene al menos una sección wifi-iface por adaptador hardware para definir una red inalámbrica.

Un ejemplo de declaración de una red wifi:

```
config 'wifi-iface'
        option 'device'         'wl0'
        option 'network'        'lan'
        option 'mode'           'ap'
        option 'ssid'           'MyWifiAP'
        option 'encryption'     'psk2'
        option 'key'            'secret passphrase'
```

- wl0 es el identificador del hardware inalámbrico.
- lan especifica la interfaz de red a la que está conectada la interfaz wifi.
- AP es el modo de operación (punto de acceso)
- MyWifiAP es el SSID que se emite.
- psk2 especifica el método de cifrado inalámbrico (WPA2-PSK).
- contraseña secreta es el secreto frase de contraseña WPA

- **Opciones Comunes**

Las opciones de configuración para la wifi-iface se enumeran a continuación.

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| device | String | sí | (Primer dispositivo id) | Especifica el adaptador inalámbrico que utiliza, debe hacer referencia a uno de los dispositivos definidos en la sección wifi |
| mode | String | sí | ap | Selecciona el modo de operación de la red inalámbrica, ap para punto de acceso, sta para modo managed (cliente), ad-hoc para Ad-Hoc y monitor para modo monitor. |
| ssid | String | sí | OpenWrt | SSID de la red inalámbrica |
| bssid | BSSID | no | (Por defecto el conductor) | Reemplaza el BSSID de la red, sólo aplicable en adhoc o modo sta. |
| hidden | Boolean | no | 0 | Si se pone a 1, se oculta el SSID |
| network | String | sí | lan | Especifica la interfaz de red a la que conectarse |
| encryption | String | no | ninguno | Se utiliza en las redes cifradas: wep para WEP, psk de WPA-PSK y psk2 para WPA2-PSK. |
| key | número o string | no | (Ninguno) | Especifica la contraseña secreta que se utiliza para WPA-PSK o WEP. |

| Nombre | Tipo | Obligatorio | Valor por defecto | Descripción |
|---|---|---|---|---|
| key1 | string | no | (Ninguno) | Clave WEP # 1 |
| key2 | string | no | (Ninguno) | Clave WEP # 2 |
| key3 | string | no | (Ninguno) | Clave WEP # 3 |
| key4 | string | no | (Ninguno) | Clave WEP # 4 |

- **WPA (Punto de Acceso)**

Opciones de WPA como punto de acceso.

| Nombre | Valor por defecto | Descripción |
|---|---|---|
| server | *(Ninguno)* | Servidor RADIUS para la gestión de clientes. |
| port | 1812 | El puerto del servidor RADIUS |
| key | *(Ninguno)* | Clave secreta de RADIUS |

## 4.3. Desarrollando para OpenWrt.

### *4.3.1. El sistema de desarrollo.*

Uno de los mayores retos para comenzar a trabajar con dispositivos integrados es que simplemente no puede instalar una copia de Linux y esperar que sea capaz de compilar un firmware. Incluso instalado un compilador y todas las herramientas de desarrollo que ofrece, aún no se tendría el conjunto básico de herramientas necesarias para producir una imagen del firmware. El dispositivo embebido representa una nueva plataforma de hardware, que es incompatible con el hardware del equipo de desarrollo, por lo que se utiliza un proceso llamado compilación cruzada para producir un nuevo compilador capaz de generar código para la plataforma embebida y también se utiliza para compilar una distribución básica de Linux que sea posible instalar en el dispositivo.

Un compilador cruzado es un compilador capaz de crear código ejecutable para otra plataforma distinta a aquélla en la que él se ejecuta. Esta herramienta es útil cuando quiere compilarse código para una plataforma a la que no se tiene acceso, o cuando es incómodo o imposible compilar en dicha plataforma (como en el caso de los sistemas empotrados).

### *4.3.2. Creando una imagen.*

OpenWrt adopta un enfoque diferente en la construcción de un firmware; descarga, parches y compilación desde el principio, incluyendo el compilador cruzado. Para ponerlo en términos más simples, OpenWrt no contiene ejecutables o incluso las fuentes, es un sistema automatizado para la descarga de las fuentes, parcheando para trabajar con la plataforma determinada y compilar correctamente para esa plataforma. Lo que esto significa es que simplemente cambiando la plantilla, puede cambiar cualquiera de las fases en el proceso.

A modo de ejemplo, si un nuevo kernel es liberado, un simple cambio a uno de los Makefiles descargará el último kernel, al ejecutar el parche se producirá una nueva imagen del firmware. Es una operación simple que permite estar siempre actualizado a OpenWrt.

Ahora procederemos a explicar cómo funciona todo este proceso:

- **Descargar OpenWrt**

Procederemos a descargarnos la versión de desarrollo desde subversión utilizando el siguiente comando:

*svn co svn://svn.openwrt.org/openwrt/branches/8.09*

- **La estructura de los directorios.**

Hay cuatro directorios clave en la base:

- tools
- toolchain
- package
- target

tools y toolchain se refieren a las herramientas comunes que se utilizarán para construir la imagen del firmware, el compilador y la biblioteca C. El resultado de esta es de tres nuevos directorios, build_dir/host, que es un directorio temporal para la construcción de las herramientas, build_dir/toolchain-<arch>* que se utiliza para la construcción de las herramientas para una arquitectura específica, y staging_dir/toolchain-<arch>*toolchain donde está instalado el archivo resultante.

- build_dir / host
- build_dir/toolchain-<arch>*

package es para alojar los paquetes de OpenWrt. En un firmware OpenWrt, casi todo es .ipk, un paquete de software que puede ser añadido al firmware para proporcionar nuevas características o eliminado para ahorrar espacio.

*$ ./scripts/feeds update*

Estos paquetes se pueden utilizar para extender la funcionalidad del sistema de compilación. Una vez hecho eso, los paquetes se mostrarán en el menú de configuración. Ejemplo para kamikaze:

*$ ./scripts/feeds search nmap*
Search results in feed 'packages':
nmap Network exploration and/or security auditing utility
*$ ./scripts/feeds install nmap*

Para incluir todos los paquetes, ejecute el siguiente comando:

*$ make package/symlinks*

target se refiere al dispositivo empotrado, este contiene elementos que son específicos de este tipo de dispositivos. De particular interés es el directorio "target/linux" que se desvincula de la plataforma <arch> y contiene los parches para el núcleo de una plataforma en particular. También está el directorio "target/image" que describe cómo empaquetar un firmware para una plataforma específica.

Todos los paquetes que se utilizan en el directorio "build_dir /<arch> " son archivos temporales que se utilizan durante la compilación.

- **Construyendo OpenWrt**

El entorno de compilación de OpenWrt está pensado para desarrolladores, aunque este es lo suficientemente simple para que un usuario sin experiencia sea capaz de compilar su propio firmware.

Al ejecutar el comando "make menuconfig" se abrirá el menú de configuración de OpenWrt, a través de este menú se puede seleccionar la plataforma para la que se está compilando y las herramientas y paquetes deseadas para construir el firmware. La selección que se haga se revisara para comprobar que cumple todas las dependencias y que el firmware sea capaz de ejecutarse sin problemas. Si la comprobación falla, se tendrán que instalar las herramientas necesarias en el sistema operativo para que se pueda realizar la compilación.

En todas las opciones de configuración se pueden elegir tres tipos de configuración:

*<*> (pulsando la tecla y)*
Este paquete se incluye en la imagen del firmware.

*<M> (pulsando la tecla m)*
Este paquete será compilado y se creara un paquete con extensión ipk para su posterior instalación.

*<> (pulsando la tecla n)*
Este paquete no se compilará.

Una vez que haya terminado con la configuración de los menús, al salir y cuando se le pida, guarde los cambios de configuración.

Para empezar a compilar el firmware, escriba "make". De forma predeterminada OpenWrt sólo mostrará una visión general de alto nivel del proceso de compilación y no de cada comando individual.

Para obtener una visión de todos los comandos que se ejecutan se deberá escribir “make V=99”

Una vez terminada la compilación del firmware, podremos encontrar las versiones para instalar en el router en la siguiente ruta: 8.09/bin/

En el apartado **4.4** de este proyecto se explicará con más detalle el funcionamiento de make menuconfig.

### *4.3.3. Creando un archivo binario capaz de ejecutarse en OpenWrt.*

En esta sección se explicará como generar el binario de un programa para que este pueda ejecutarse en el sistema, este no es el caso idóneo, ya que es más sencillo para los usuarios finales el crear un paquete que estos puedan instalar sin tener que preocuparse de tener que realizar todas las operaciones que se explican a continuación.

Aunque esta forma de compilar un programa sea poco recomendada, nos ayudará a entender todas las variables y las herramientas que entran en juego en la compilación cruzada.

Para explicar este proceso, supondremos que tenemos descargado y compilada la versión de desarrollo de OpenWrt en el directorio home (~/) de nuestro usuario.

Lo primero que tenemos que hacer es añadir al PATH del sistema el directorio donde se encuentra el compilador cruzado:

*PATH=$PATH:"8.09/staging_dir/toolchain-mipsel_gcc3.4.6/bin"*

Lo siguiente será indicar el compilador de C que vamos a utilizar:

*CC=mipsel-linux-uclibc-gcc*

Por último, se procederá a compilar el programa, aquí existen dos casos:

- **Creando Makefile con un archivo configure:**

*./configure --host=mipsel-linux*

- **Creando el archivo ejecutable directamente:**

*mipsel-linux-uclibc-gcc helloworld.c –o helloworld*

Una vez completado este proceso, se nos creara un archivo binario el cual se podrá ejecutar en el router.

### *4.3.4. Creando paquetes para OpenWrt.*

Una de las metas del software de desarrollo de OpenWrt es que sea muy fácil portar aplicaciones a esta plataforma. Si nos fijamos en el directorio de un paquete para OpenWrt encontraremos estas cosas:

- package/<name>/Makefile
- package/<name>/patches
- package/<name>/files

El directorio de patches es opcional y normalmente contiene correcciones de errores u optimizaciones para reducir el tamaño del ejecutable. El archivo Makefile es el más importante ya que proporciona los pasos realmente necesarios para descargar y compilar el paquete.

El directorio files es también opcional y contiene scripts de inicio y configuraciones por defecto del paquete que se va a utilizar en OpenWrt.

Al mirar el contenido de un archivo Makefile, difícilmente vamos a identificarlo como este, ya que este archivo se ha convertido en una plantilla orientada al simplificar la creación de paquetes.

A continuación se muestra el Makefile del paquete bridge, package/bridge/Makefile:

```
1  # $Id: Makefile 5624 2006-11-23 00:29:07Z nbd $
2
3  include $(TOPDIR)/rules.mk
4
5  PKG_NAME:=bridge
6  PKG_VERSION:=1.0.6
7  PKG_RELEASE:=1
8
9  PKG_SOURCE:=bridge-utils-$(PKG_VERSION).tar.gz
10 PKG_SOURCE_URL:=@SF/bridge
11 PKG_MD5SUM:=9b7dc52656f5cbec846a7ba3299f73bd
12 PKG_CAT:=zcat
13
14 PKG_BUILD_DIR:=$(BUILD_DIR)/bridge-utils-$(PKG_VERSION)
15
16 include $(INCLUDE_DIR)/package.mk
17
18 define Package/bridge
19   SECTION:=net
20   CATEGORY:=Base system
21   TITLE:=Ethernet bridging configuration utility
22   URL:=http://bridge.sourceforge.net/
23 endef
24
25 define Package/bridge/description
26   Manage ethernet bridging:
27   a way to connect networks together to form a larger network.
28 endef
29
30 define Build/Configure
31     $(call Build/Configure/Default, \
32         --with-linux-headers="$(LINUX_DIR)" \
33     )
34 endef
35
36 define Package/bridge/install
37     $(INSTALL_DIR) $(1)/usr/sbin
38     $(INSTALL_BIN) $(PKG_BUILD_DIR)/brctl/brctl $(1)/usr/sbin/
39 endef
40
41 $(eval $(call BuildPackage,bridge))
```

Como se puede observar, la implementación es muy sencilla y solo nos quedara declarar algunas variables.

- PKG_NAME
  El nombre del paquete, como se ve en menuconfig e ipkg
- PKG_VERSION
  La versión que se descarga
- PKG_RELEASE
  La versión de este Makefile
- PKG_SOURCE
  El nombre de archivo de las fuentes originales
- PKG_SOURCE_URL
  URL desde la cual se puede descargar el código fuente
- PKG_MD5SUM
  Una suma md5 para validar la descarga
- PKG_CAT
  Cómo descomprimirlos ficheros
- PKG_BUILD_DIR
  Donde compilar el paquete

Las variables PKG_* definen dónde descargar el paquete desde; @SF es una clave especial para la descarga de paquetes desde sourceforge. También hay otra palabra clave de @GNU para descargar las fuentes de GNU. Si alguna de la fuente de descarga falla, se utilizaran los archivos de la web de OpenWrt.

El md5sum se utiliza para verificar que el paquete se ha descargado correctamente y PKG_BUILD_DIR define dónde encontrar las fuentes descomprimidas en $(BUILD_DIR).

En la parte inferior del fichero es donde realmente se compila el paquete "buildpackage" es una macro creada para incluir las declaraciones. Buildpackage sólo toma un argumento directamente - el nombre del paquete que se construirá. Cualquier otra información es tomada de la definición de los bloques. Esta es una forma de proporcionar un nivel alto de abstracción.

Buildpackage utiliza lo siguiente:

- *Package/<name>:*
  <name> coincide con el argumento pasado a buildroot, este describe el paquete en menuconfig y el paquete opkg. Dentro de Package/<name> puede definir las siguientes variables:

  - SECCIÓN
    El tipo de paquete (no se utiliza)
  - CATEGORÍA
    En que menú del menuconfig aparece: red, sonido, utilidades, multimedia ...
  - TÍTULO
    Una descripción corta del paquete
  - URL
    Dónde encontrar el software original

- MAINTANER (opcional)
  A quien contactar sobre el paquete
- DEPENDS (opcional)
  Qué paquetes se deben compilar/instalar antes de este paquete.
- BUILDONLY (opcional)
  Establezca esta opción a 1 si no desea que su paquete aparezca en menuconfig. Esto es útil para paquetes que sólo se utilizan como dependencias de compilación.

- *Package/<name>/conffiles (opcional):*
  Una lista de los ficheros de configuración instalados con este paquete, un archivo por línea.

- *Build/Prepare (opcional):*
  Un conjunto de comandos para desempaquetar y parchear el código fuente.

- *Build/Configure (opcional):*
  Puede omitirse en caso de que no tengamos archivo de configuración o el archivo de configuración sea simple, para el resto de casos se pueden escribir comandos personalizados o usar "$(call Build/Configure/Default,<first list of arguments, second list>)".

Para que sea más sencillo modificar la configuración de la línea de comandos, se pueden añadir o quitar las siguientes variables:

- CONFIGURE_ARGS
  Contiene todos los argumentos de la línea de comandos (formato: arg-1-2 arg)
- CONFIGURE_VARS
  Contiene todas las variables de entorno que se pasan a ./configure (formato: NAME = "valor")

- *Build/Compile (opcional):*
  Cómo compilar las fuentes, en la mayoría de los se omite este apartado.

Para que sea más sencillo modificar la configuración de la línea de comandos, se pueden añadir o quitar las siguientes variables:

- MAKE_FLAGS
  Contiene todos los argumentos de la línea de comandos

- MAKE_VARS
  Contiene todas las variables de entorno que se pasan al comando make

- *Package/<name>/install:*

Un conjunto de comandos para copiar archivos de la fuente.

Después de haber creado el archivo package/<name>/Makefile, el nuevo paquete se mostrarán automáticamente en el menú la próxima vez que ejecute "make menuconfig"

y si se selecciona será compilado la próxima vez que se ejecute make o con el comando make package/<name>/compile.

### 4.4. Make menuconfig.

make menuconfig es una de las herramientas que se utilizan para configurar el kernel de Linux, un paso necesario antes de compilar el código fuente. make menuconfig, posee una interfaz de usuario sencilla, permite al usuario elegir las características del kernel de Linux (y otras opciones). Se invoca mediante el comando make menuconfig.

#### *4.4.1. Dependencias.*

Para poder utilizar make menuconfig, es necesario el código fuente del kernel de Linux, la herramienta make, un compilador de C y la librería ncurses.

#### *4.4.2. Teclas útiles.*

| Tecla | Significado |
|---|---|
| ¿ | Descripción de opciones y consejos o ayudas |
| Left/Right (↔) Up/Down (↕) PgUp/PgDn | Navegar por el menú |
| Esc | Salir del menuconfig o cancelar el comando |
| Enter (⏎) | Activar un comando o desplegar un menú |
| y | Compilar e incluir este paquete en el kernel |
| m | Compilar como paquete, sin incluir en el kernel |
| n | No compilar |

### *4.4.3. Símbolos*

| Símbolo | Significado |
|---|---|
| <> | Sin dependencias |
| [] | Una dependencia indica que se compile (y) o no (n) |
| {} | Una dependencia indica que es un modulo (m) o que se compile (y) |
| -- | Una dependencia indica que esta compilado (y) |

La información suministrada es la dependencia primaria, no dice los nombres de las dependencias.

| Símbolo | Significado |
|---|---|
| → | Utilice entrar (⏎) para expandir esta rama como una nueva ventana. |
| (Experimental) | Código inestable |
| (Nuevo) | Una versión nueva del paquete. |

### *4.4.4. Descripción de los diferentes menús de make menuconfig.*

Al ejecutar el comando make menuconfig, nos encontraremos con el siguiente menú:

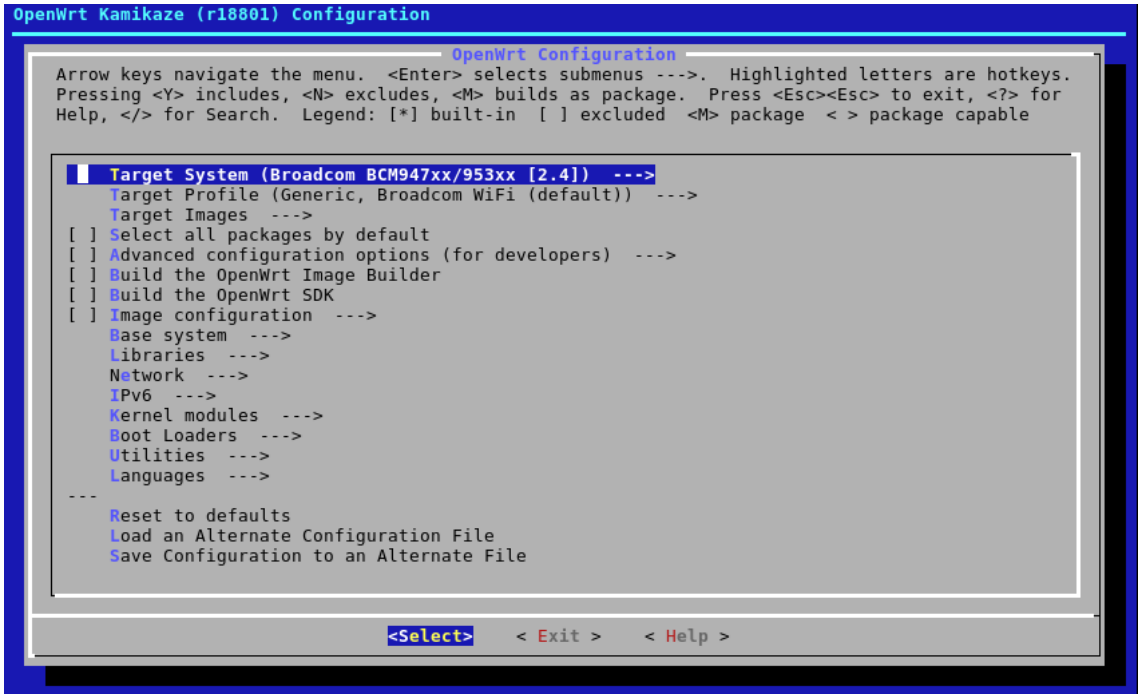



**Imagen 4. 3: Menú principal menuconfig**

En este menú se muestran las opciones principales de configuración y las distintas categorías de las herramientas y paquetes disponibles para OpenWrt. A continuación se dará una pequeña explicación del significado de las distintas opciones:

- *Tarjet System (Broadcom BCM947xx/953xx [2.4]):* en este apartado se puede seleccionar entre los distintos chips y kernel para los que está preparado OpenWrt. Esta selección es muy importante, ya que si no seleccionamos el chip y kernel correcto, ni el firmware, ni los paquetes, ni nada de lo que se compile será capaz de correr en el router.

- *Tarjet Profile (Generic, Broadcom Wifi (default)):* en este menú se seleccionara el driver wifi que se utilizara en nuestro router.

- *Tarjet Images:* nos permite seleccionar el sistema de archivo y la compresión que se va a utilizar en nuestro firmware.

- *Select all packages by default:* esta opción selecciona todos los paquetes para que se compilen, esta opción es útil ya que nos compilara todos los paquetes disponibles y no tendremos que estar compilando cada paquete en caso de necesidad.

- *Advanced configuration options (for developers):* esta opción habilita ciertas opciones del kernel de OpenWrt para que a los desarrolladores les resulte más sencillo realizar modificaciones en el kernel de OpenWrt.

- *Build the OpenWrt Image Builder:* esta opción nos permitirá compilar el Image Builder de OpenWrt.
- *Build the OpenWrt SDK:* esta opción nos permitirá compilar el SDK de OpenWrt.
- *Image configuration:*
- *Base System, Libraries, Network, IPv6, kernel modules, Boot Loaders, Utilities, Languajes:* todas estas opciones hacen referencia a las distintas categorías en las que se encuentran los paquetes que podemos seleccionar de OpenWrt.
- *Reset to defaults:* pone todas las opciones de OpenWrt a sus valores por defecto.
- *Load an Alternate Configuration File:* si tenemos guardado algún archivo de configuración del menuconfig, con esta opción podemos volver a cargarlo.
- *Save Configuration to an Alternate File:* para guardar el archivo de configuración con otro nombre y/o ubicación.

A continuación les daremos una pequeña introducción sobre cada submenú.

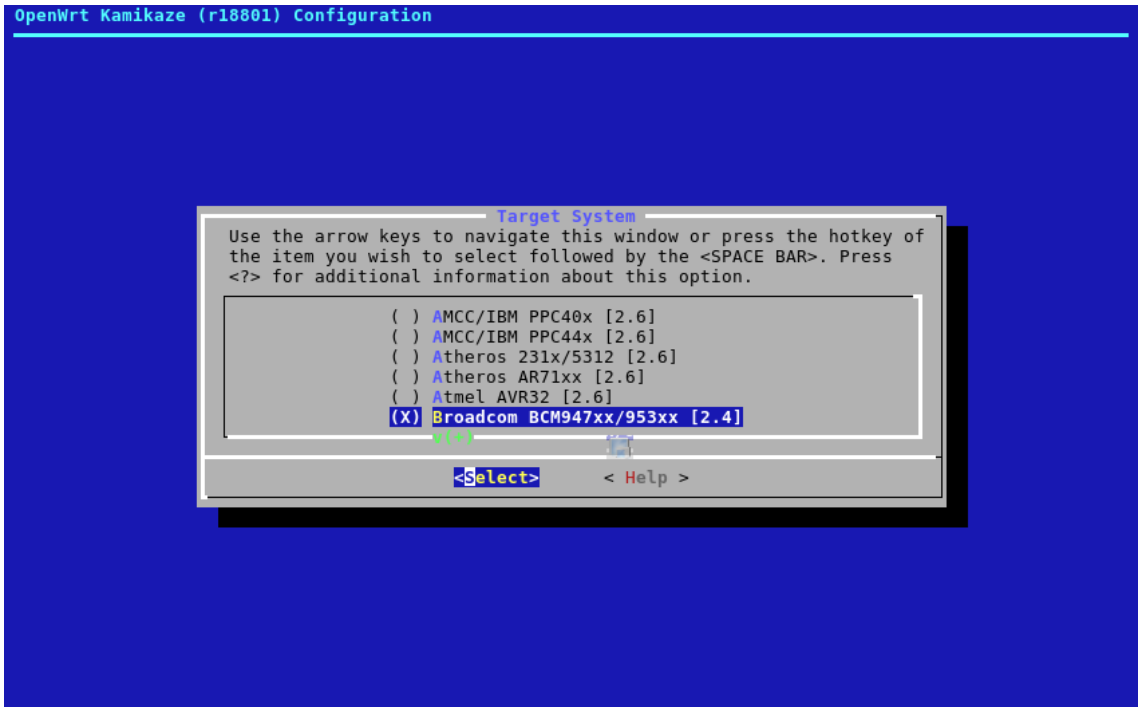



**Imagen 4. 4: Target System**

En este submenú se puede seleccionar el chip y el kernel de nuestro sistema, aquí aparecen una pequeña muestra de la combinación de chip y kernel a seleccionar.

Ahora mismo está seleccionado Broadcom BSC947xx/953xx [2.4], de este nombre podemos sacar la siguiente información:

- *Chip Broadcom BCM947xx/953xx*
- *Kernel de Linux que se utiliza 2.4*

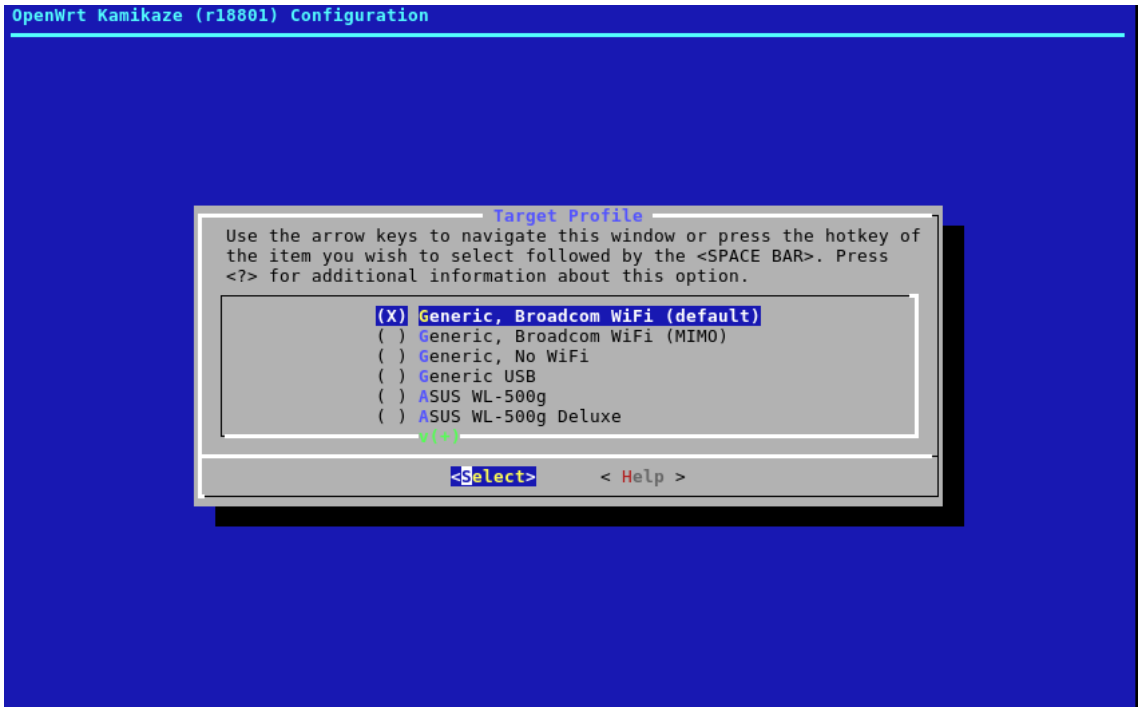



**Imagen 4. 5: Target Profile**

En esta sección podemos seleccionar el driver que utilizara nuestro sistema, la información que se muestra es el tipo de driver y el chip al que va destinado.

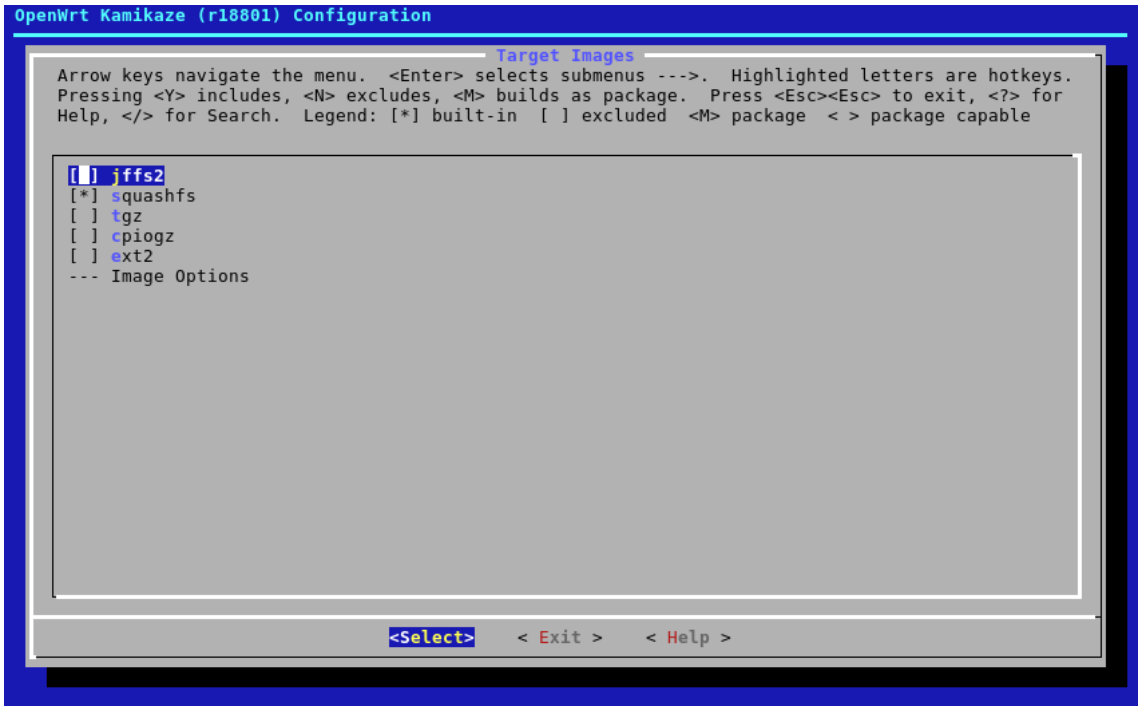



**Imagen 4. 6: Tarjet Images**

En este submenú, seleccionaremos el sistema de ficheros y la compresión que se utilizara en nuestro sistema. La opción que se utilizará casi siempre es la de squashfs, ya que esta opción nos da la posibilidad de utilizar la recuperación del sistema en caso de que sea necesaria, debido a que es la única opción que crea dos particiones reales, la del sistema y la de recuperación.

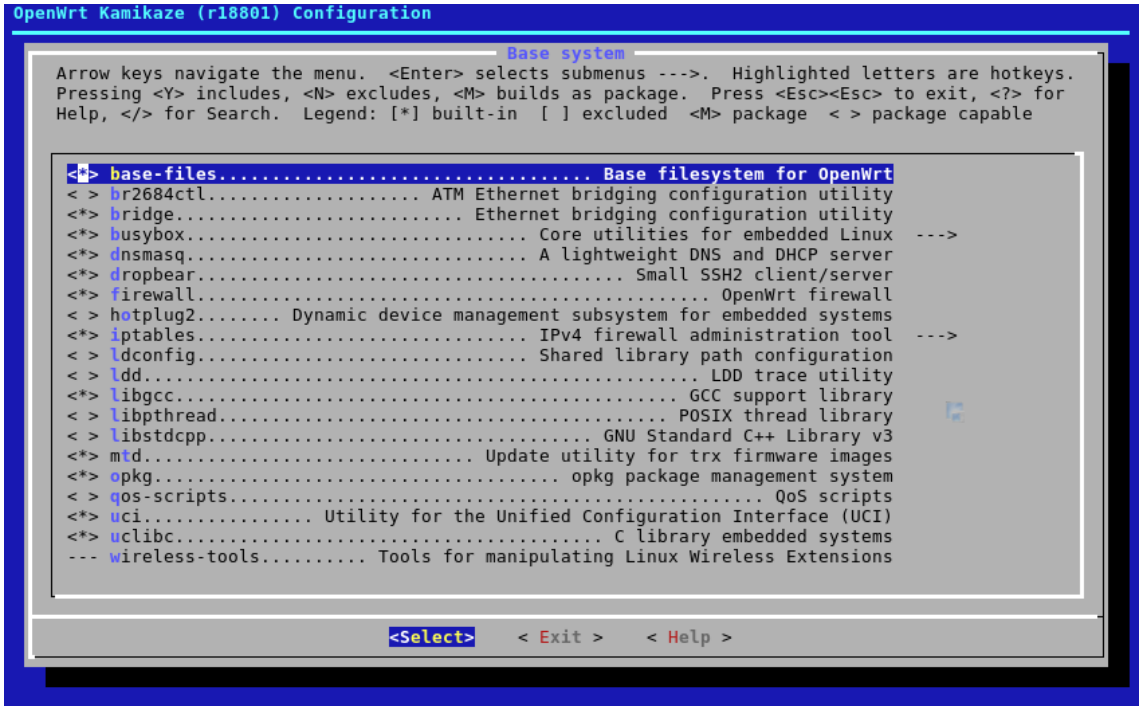



**Imagen 4. 7: Base System**

Como podemos observar en la configuración por defecto, están seleccionados los paquetes mínimos para que el sistema sea capaz de funcionar correctamente.

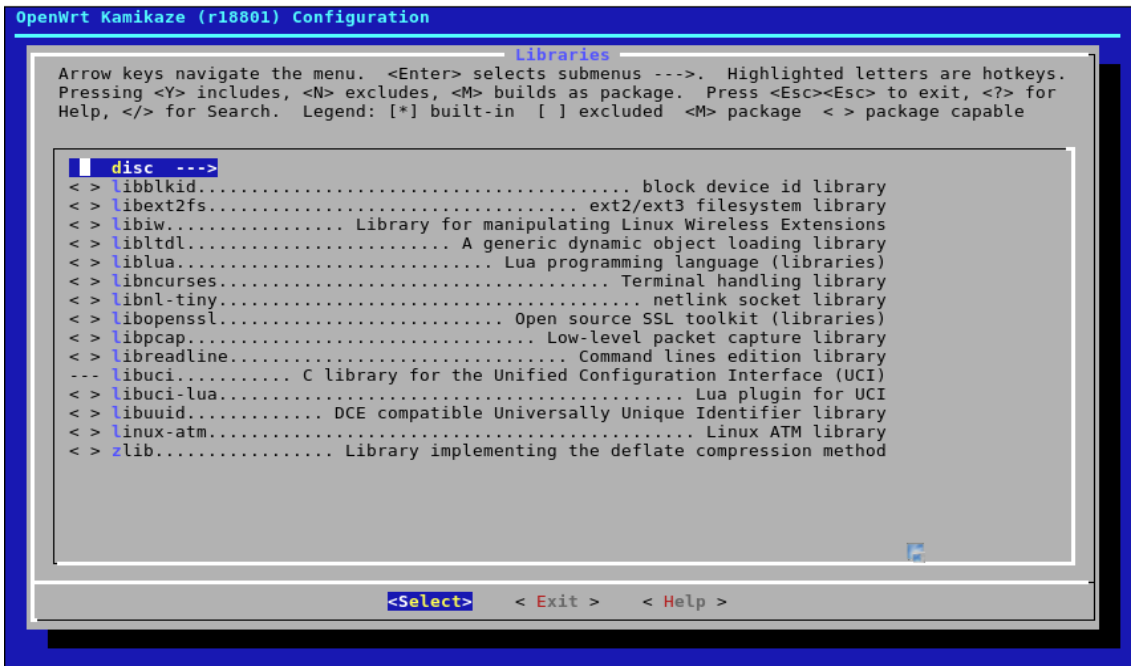



**Imagen 4. 8: Libraries**

En este submenú se encuentran las librerías del kernel de Linux, estas librerías son las que extienden las funciones del kernel.

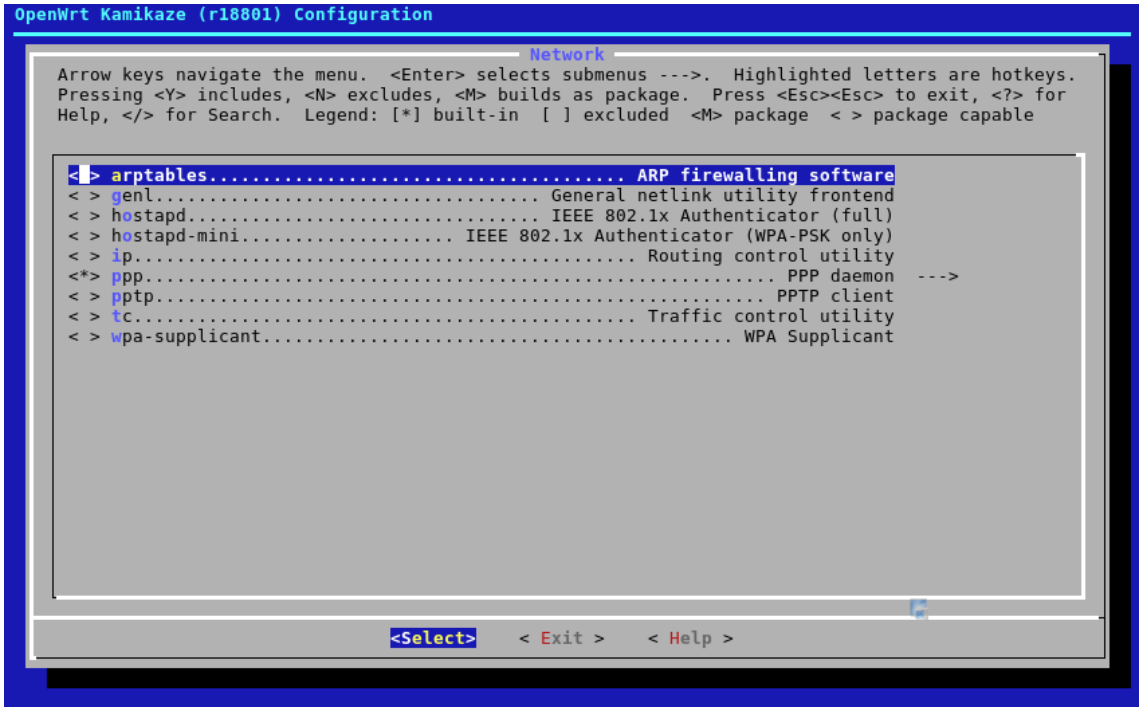



**Imagen 4. 9: Network**

En este submenú se pueden añadir paquetes para aumentar las funcionalidades de la red del sistema.

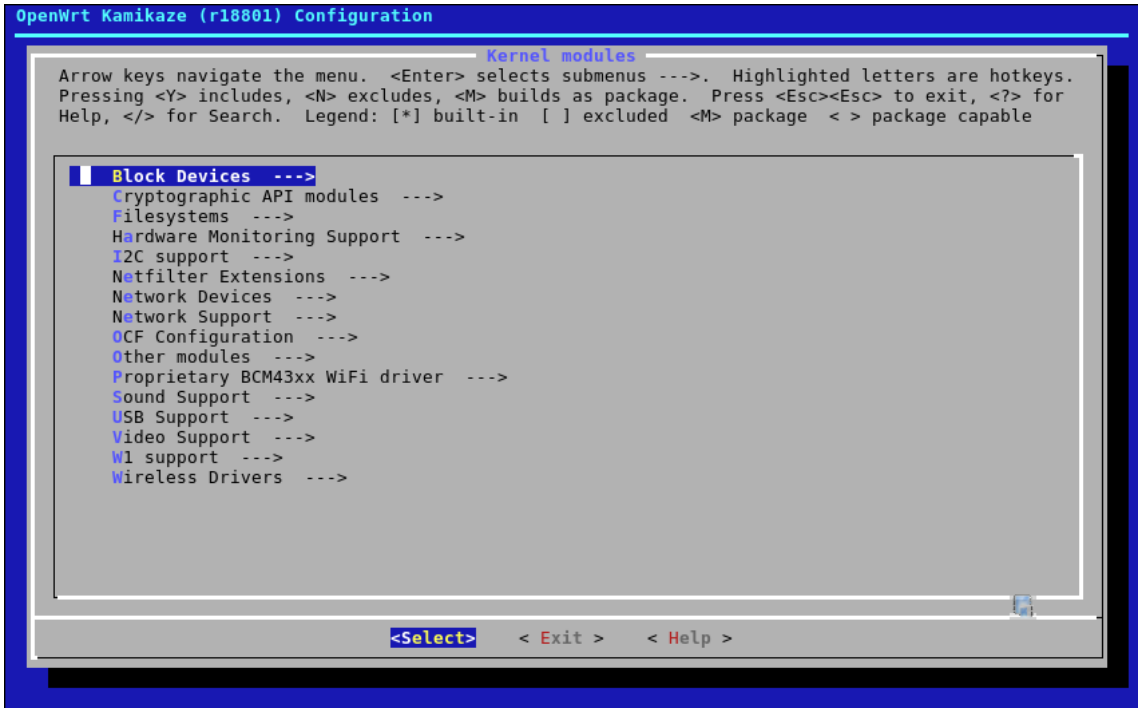



**Imagen 4. 10: kernel modules**

En este submenú, se podrán seleccionar paquetes para aumentar la funcionalidad de las diferentes partes del sistema.

# 5. Implementación del sistema final.

Para la implementación de este proyecto, será necesaria la utilización de un ordenador que disponga de un sistema operativo basado en Linux, nosotros utilizaremos la distribución de Linux conocida como Ubuntu. Si no dispone de una copia de Ubuntu para la realización de este proyecto, puede adquirir o descargarse su copia desde la web de Ubuntu (http://www.ubuntu.com/).

El uso del sistema Linux será imprescindible para poder completar este proyecto fin de carrera, aunque en los casos en los que sea posible utilizar otro sistema operativo distinto a linux, se explicara el procedimiento a seguir en caso de ser distinto.

## 5.1. Descargando las fuentes.

Lo primero que debemos hacer es comprobar si tenemos instalada la herramienta subversión (svn) en nuestro sistema operativo basado en Linux. En caso de no tenerla instalada, procederemos a instalarla:

*sudo aptitude install subversión*

Una vez instalada la herramienta subversión, procederemos a descargarnos los archivos fuentes desde el repositorio de OpenWrt:

*svn co svn://svn.openwrt.org/openwrt/branches/8.09*

```
paco@kubuntu:~/PFC2$ svn co svn://svn.openwrt.org/openwrt/branches/8.09
A    8.09/rules.mk
A    8.09/toolchain
A    8.09/toolchain/info.mk
A    8.09/toolchain/uClibc
A    8.09/toolchain/uClibc/config
A    8.09/toolchain/uClibc/config/arm
A    8.09/toolchain/uClibc/config/powerpc
A    8.09/toolchain/uClibc/config/cris
A    8.09/toolchain/uClibc/config/arm.storm
A    8.09/toolchain/uClibc/config/armeb
A    8.09/toolchain/uClibc/config/mips
A    8.09/toolchain/uClibc/config/i386
A    8.09/toolchain/uClibc/config/mipsel
A    8.09/toolchain/uClibc/config/i686
A    8.09/toolchain/uClibc/config/avr32
A    8.09/toolchain/uClibc/config/0.9.28.2
A    8.09/toolchain/uClibc/config/0.9.28.2/arm
A    8.09/toolchain/uClibc/config/0.9.28.2/powerpc
A    8.09/toolchain/uClibc/config/0.9.28.2/cris
A    8.09/toolchain/uClibc/config/0.9.28.2/armeb
A    8.09/toolchain/uClibc/config/0.9.28.2/mips
A    8.09/toolchain/uClibc/config/0.9.28.2/i386
A    8.09/toolchain/uClibc/config/0.9.28.2/mipsel
A    8.09/toolchain/uClibc/config/0.9.28.2/avr32
A    8.09/toolchain/uClibc/config/0.9.28.2/x86_64
A    8.09/toolchain/uClibc/config/x86_64
A    8.09/toolchain/uClibc/patches
A    8.09/toolchain/uClibc/patches/140-fix-endless-recursion-in-pthread.patch
A    8.09/toolchain/uClibc/patches/110-compat_macros.patch
A    8.09/toolchain/uClibc/patches/130-compile_fixes.patch
A    8.09/toolchain/uClibc/patches/006-rm_whitespace.patch
```

**Imagen 5. 1: Descarga versión de desarrollo**

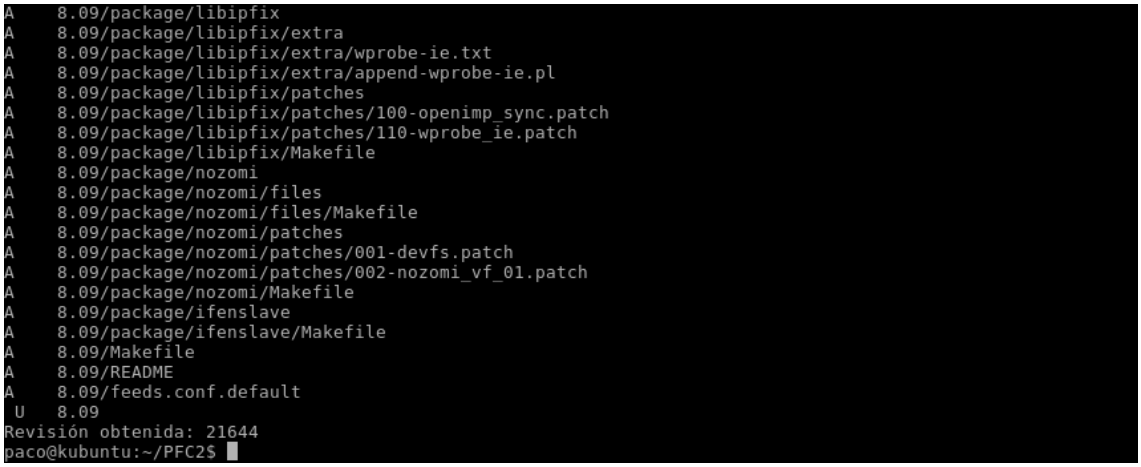



**Imagen 5. 2: Revisión obtenida: 21644**

Al terminar la descarga de las fuentes podemos observar cómo se nos muestra la revisión descargada de las fuente, en nuestro caso es la revisión 21644, una vez obtenidas las fuentes, podemos observar como ahora tenemos una nueva carpeta llamada *8.09*, en el interior de esta carpeta se encuentran todas las herramientas necesarias para la realización de este proyecto.


**Imagen 5. 3: Directorio de la versión de desarrollo.**

## 5.2. Configurando make menuconfig.

Una vez comprobado que se nos ha creado el directorio 8.09, debemos acceder a dicha carpeta con el siguiente comando:

*cd 8.09/*

Podemos comprobar, todos los archivos y carpetas que se nos han creado con el comando de consola:

*ls*

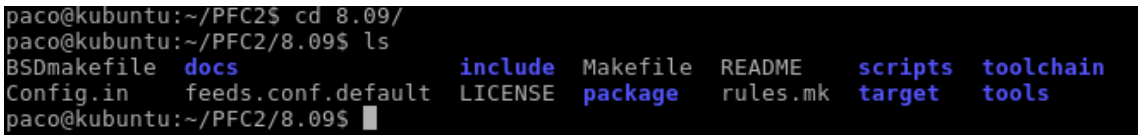



**Imagen 5. 4: Directorios de la versión de desarrollo**

Una vez comprobados los directorios, procederemos a ejecutar el siguiente comando para configurar nuestro sistema:

*make menuconfig*

Nada más ejecutar este comando, se harán una serie de precomprobaciones para ver si la configuración del sistema operativo es la correcta para poder utilizar el

menuconfig, en caso de que alguna prueba no tenga éxito, tendremos que solucionarla antes de poder continuar con el proyecto:

```
paco@kubuntu:~/PFC2/8.09$ make menuconfig
Checking 'working-make'... ok.
Checking 'case-sensitive-fs'... ok.
Checking 'getopt'... ok.
Checking 'fileutils'... ok.
Checking 'working-gcc'... ok.
Checking 'working-g++'... ok.
Checking 'ncurses'... ok.
Checking 'zlib'... ok.
Checking 'gawk'... ok.
Checking 'flex'... ok.
Checking 'unzip'... ok.
Checking 'bzip2'... ok.
Checking 'patch'... ok.
Checking 'perl'... ok.
Checking 'python'... ok.
Checking 'wget'... ok.
Checking 'gnutar'... ok.
Checking 'svn'... ok.
Checking 'gnu-find'... ok.
```

**Imagen 5. 5: make menuconfig**

Una vez que han terminado las comprobaciones, aparecerá el menú principal de menuconfig. A continuación se enumeraran las opciones que debemos seleccionar para poder continuar con este proyecto.

En la siguiente captura, hemos seleccionado todos los paquetes por defecto, la configuración para desarrolladores y que compile el SDK.

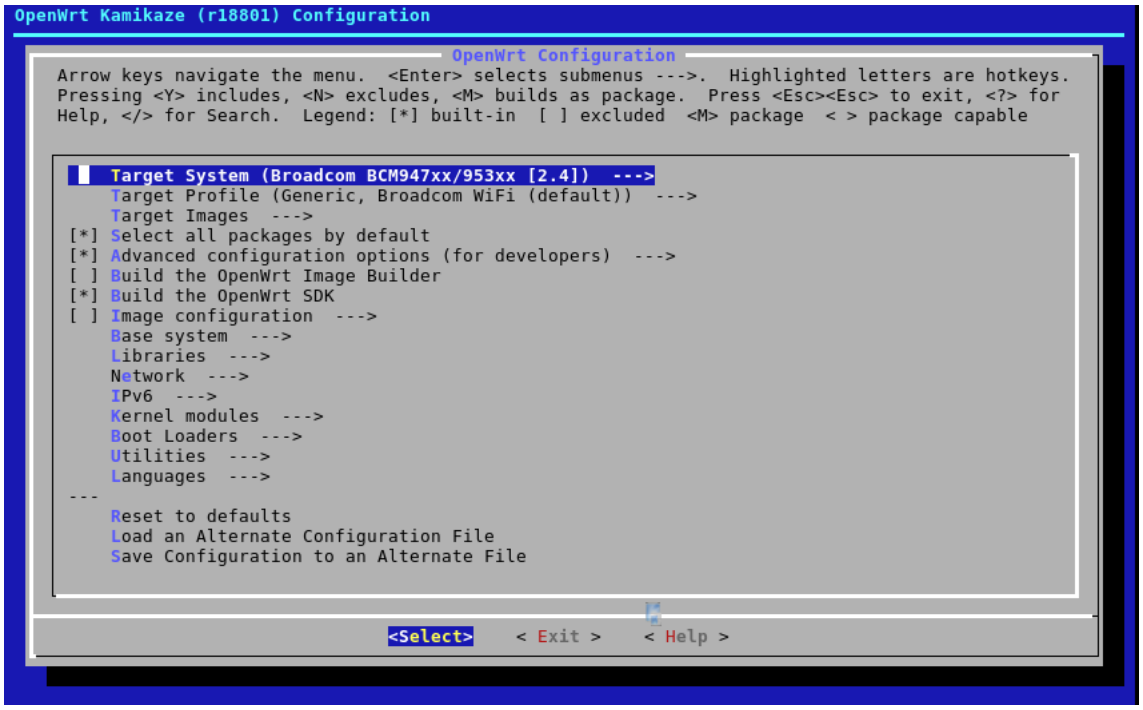



**Imagen 5. 6: menuconfig**

Todas las demás opciones las dejaremos tal cual vienen por defecto, excepto dentro del submenú Libraries, en el que seleccionaremos que la librería de libpcap se inserte automáticamente en el firmware generado. Podemos observar como quedaría la configuración en la siguiente imagen.

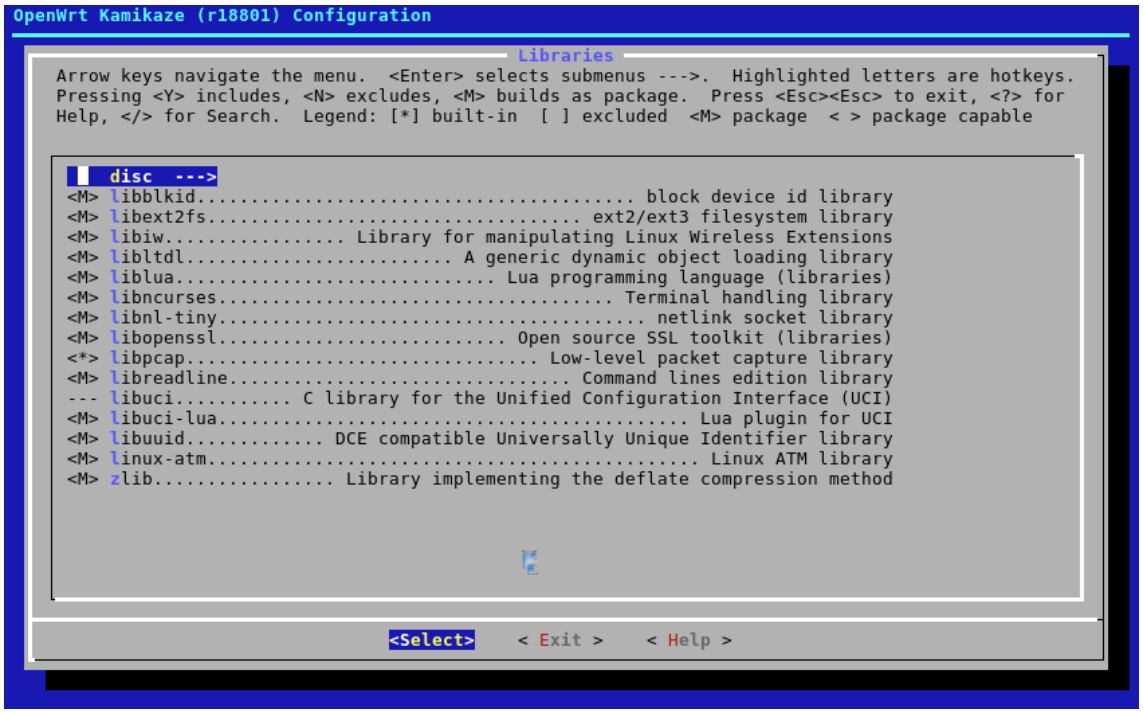



**Imagen 5. 7: Selección de la librería libpcap**

Finalmente procederemos a guardar todos los cambios realizados al salir de menuconfig, para después proceder a la compilación.

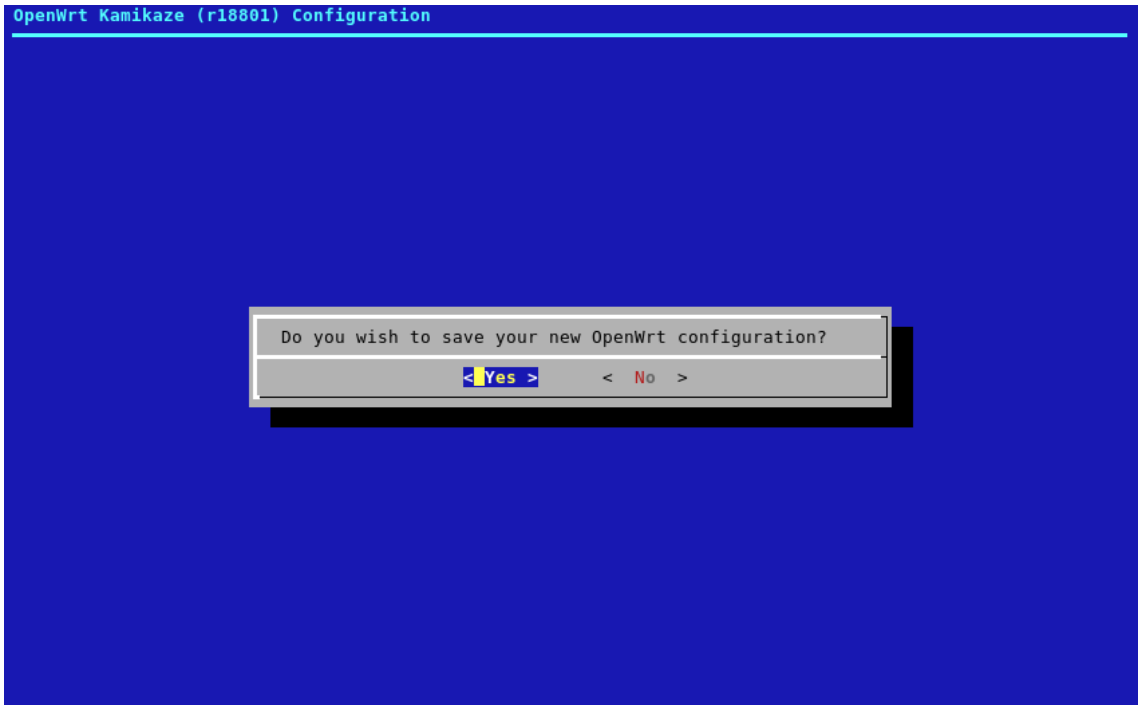



**Imagen 5. 8: guardando el archivo de configuración**

Cuando salgamos de menuconfig, ejecutaremos el siguiente comando para comenzar el proceso:

*make*

En caso de que se produzca algún error, podemos ejecutar el siguiente comando, para observar el error e intentar corregirlo.

*make V=99*

```
paco@kubuntu:~/PFC2/8.09$ make
++ mkdir -p /home/paco/PFC2/8.09/staging_dir/toolchain-mipsel_gcc3.4.6
++ cd /home/paco/PFC2/8.09/staging_dir/toolchain-mipsel_gcc3.4.6
++ mkdir -p bin lib include stamp
 make[1] world
 make[2] tools/install
 make[3] -C tools/sed compile
 make[3] -C tools/sed install
 make[3] -C tools/sstrip compile
 make[3] -C tools/sstrip install
 make[3] -C tools/ipkg-utils compile
 make[3] -C tools/ipkg-utils install
 make[3] -C tools/genext2fs compile
```

**Imagen 5. 9: Comienzo del make**

Una vez terminado el largo make, ya podemos continuar con el proyecto.

### 5.3. Seleccionando e instalando OpenWrt.

Una vez terminado el proceso anterior, se nos habrán generado los archivos del firmware y todos los paquetes que hayamos seleccionado.

A continuación nos desplazaremos a una nueva carpeta que se nos ha creado llamada bin:

*cd bin/*

En el interior de esta carpeta encontraremos varios archivos de firmware y una carpeta llamada package. Con estos archivos ya podemos proceder a instalar el firmware, el proceso de instalación dependerá de la configuración actual del router, ya que tendremos que utilizar distintos procedimientos dependiendo de si tenemos el router con el firmware del fabricante o con una versión anterior de OpenWrt.

```
paco@kubuntu:~/PFC$ cd 8.09/bin/
paco@kubuntu:~/PFC/8.09/bin$ ls
md5sums                                    openwrt-wrt350n_v1-squashfs.bin
openwrt-brcm-2.4-squashfs.trx              openwrt-wrt54g3g-em-squashfs.bin
OpenWrt-SDK-brcm-2.4-for-Linux-x86_64.tar.bz2  openwrt-wrt54g3g-squashfs.bin
openwrt-usr5461-squashfs.bin               openwrt-wrt54g-squashfs.bin
openwrt-wa840g-squashfs.bin                openwrt-wrt54gs-squashfs.bin
openwrt-we800g-squashfs.bin                openwrt-wrt54gs_v4-squashfs.bin
openwrt-wr850g-squashfs.bin                openwrt-wrtsl54gs-squashfs.bin
openwrt-wrt150n-squashfs.bin               packages
openwrt-wrt300n_v1-squashfs.bin
```

**Imagen 5. 10: firmwares creados**

Para instalar el firmware tenemos las siguientes opciones:

1. Desde la página web del router, en la sección de Actualizar firmware.
2. Usando el protocolo tftp.
3. Usando el cable de serie CFE.
4. Usando JTAG.
5. Desde la línea de comandos de OpenWrt (este caso sólo si ya había una versión previa.

La forma más sencilla es la primera opción (vía web del router) y la más desaconsejada es la opción cuarta (por JTAG) ya que se puede tardar hasta dos horas para conseguir que se instale el firmware de manera correcta.

De las otras opciones hay que destacar la segunda opción, ya que si por alguna razón hubo algún problema al instalar el firmware, o no arranca de manera correcta, se podrá reinstalar el firmware a través de un cliente tftp.

Los pasos para instalar el firmware por medio de un cliente tftp es prácticamente similar en cualquier sistema operativo. En cualquiera de los tres sistemas, se tendrá que configurar el ordenador con la IP estática 192.168.1.10 y con máscara 255.255.255.0.

Para usar el cliente TFTP se verá cómo hacerlo en cada sistema:

a) Instalación desde tftp en Linux/BSD:

El router tiene que estar apagado, es decir, el cable de alimentación debe estar desconectado. Se accederá a la ruta en donde se encuentra el firmware OpenWrt descargado desde la consola de comandos, y se ejecutarán los siguientes comandos: tftp 192.168.1.1

Binary

rexmt 1

timeout 60

trace

put openwrt-xxx-x.x-xxx.bin

La opción rexmt 1 se utiliza para que el cliente reintente constantemente enviar el archivo a la dirección dada. Una vez introducidos los comandos, se conectará a la corriente el router y cuando el cargador de arranque del router comience a esperar el envío del firmware, automáticamente se le enviará. El proceso tarda un tiempo y en ningún momento se debe apagar el router. Cuando el firmware esté actualizado el router se reiniciará de nuevo.

b) Instalación desde tftp en Mac OS X.

Para Mac OS se recomienda usar el cliente MacTFTP en puesto del cliente tftp que viene integrado en el sistema, debido a ciertos fallos que han sido reportados por algunos usuarios.

1) Descargar MacTFTP:

   http://www.mactechnologies.com/pages/downld.html

2) Elegir la opción de envío.

3) Introducir la dirección del router: 192.168.1.1

4) Seleccionar el archivo que se va a enviar: openwrt-xxx-x.x-xxx.bin

5) Y finalmente pulsar en iniciar y conectar el router a la corriente.

c) Instalación desde tftp en Windows 2000/XP/Vista/7.

Habrá que desactivar cualquier cortafuegos que se tenga instalado para poder enviar el firmware a través del cliente tftp.

Los pasos para instalar son los siguientes:

1) Abrir dos consolas de comandos de Windows.

2) En la primera ejecutar: ping -t -w 10 192.168.1.1

3) El ping estará intentando conectar con el router cada 10ms constantemente.

4) En la segunda acceder a la ruta donde se encuentra el firmware OpenWrt.

5) Y ahora escribir: tftp -i 192.168.1.1 PUT openwrt-xxx-x.x-xxx.bin pero no presionar la tecla Intro (Enter).

6) Conectar el router y se mostrará en la primera ventana que aparece "Error de Hardware".

7) Tan pronto como vuelve a funcionar el ping, se ejecuta el comando de tftp de la segunda ventana y se enviará el archivo al router.

Una vez instalado el firmware, accederemos al router por medio de telnet y le asignaremos una contraseña para poder acceder por SSH:

*telnet 192.168.1.1*
*passwd*

### 5.4. Accediendo mediante SSH a la línea de comandos de OpenWrt.

Una vez instalado el firmware y configurada la contraseña para acceder a la línea de comandos, vamos a mostrar como acceder mediante SSH desde Windows y Ubuntu:

a) Accediendo mediante SSH desde Windows:

1) Necesitaremos descargarnos e instalar SSH Secure Shell client desde la web http://www.ssh.com/

2) Cuando la instalación haya finalizado, ejecutaremos Secure Shell client y nos aparecerá la siguiente ventana:

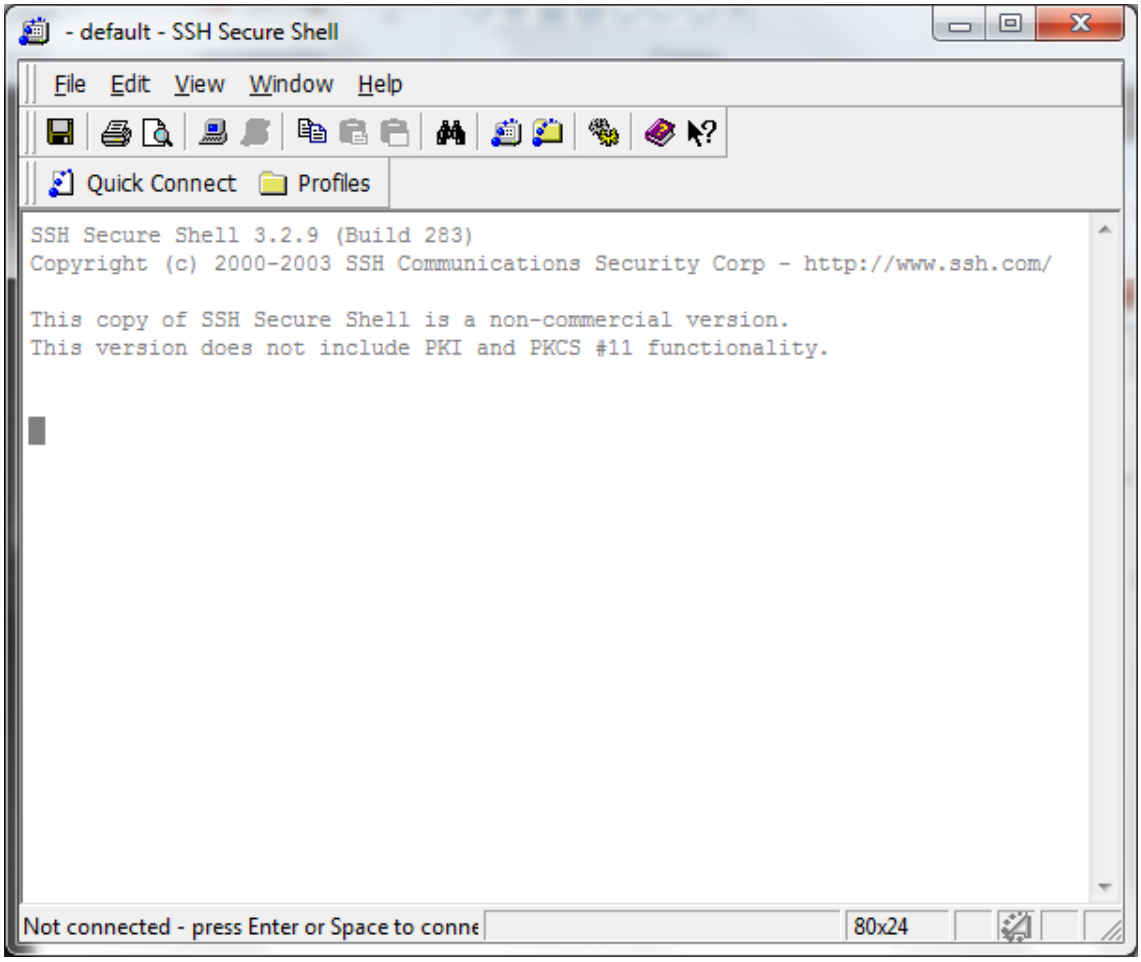



**Imagen 5. 11: Secure Shell client**

3) En la ventana anterior, deberemos pulsar sobre "Quick Connect" y nos aparecerá la siguiente ventana que hay que rellenar con los datos que se muestran:

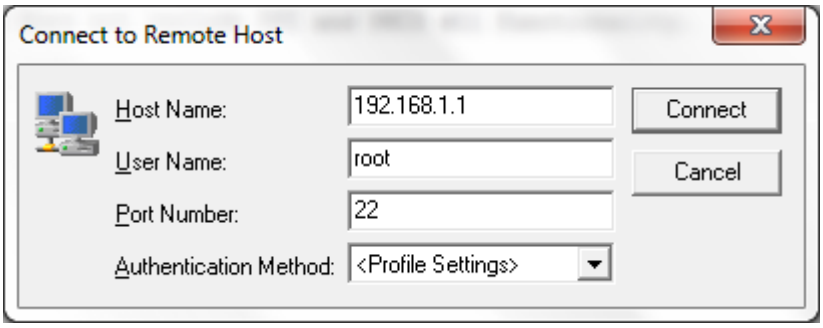



**Imagen 5. 12: Datos de conexión**

4) Una vez introducidos los datos, pulsaremos sobre “Connect” y nos aparecerá la siguiente ventana solicitándonos el password introducido anteriormente mediante telnet.

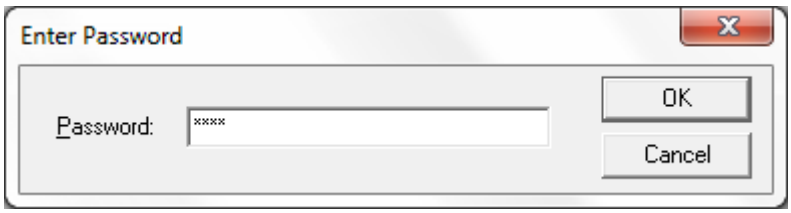



**Imagen 5. 13: password**

5) Y por ultimo nos aparecerá la línea de comandos de OpenWrt:

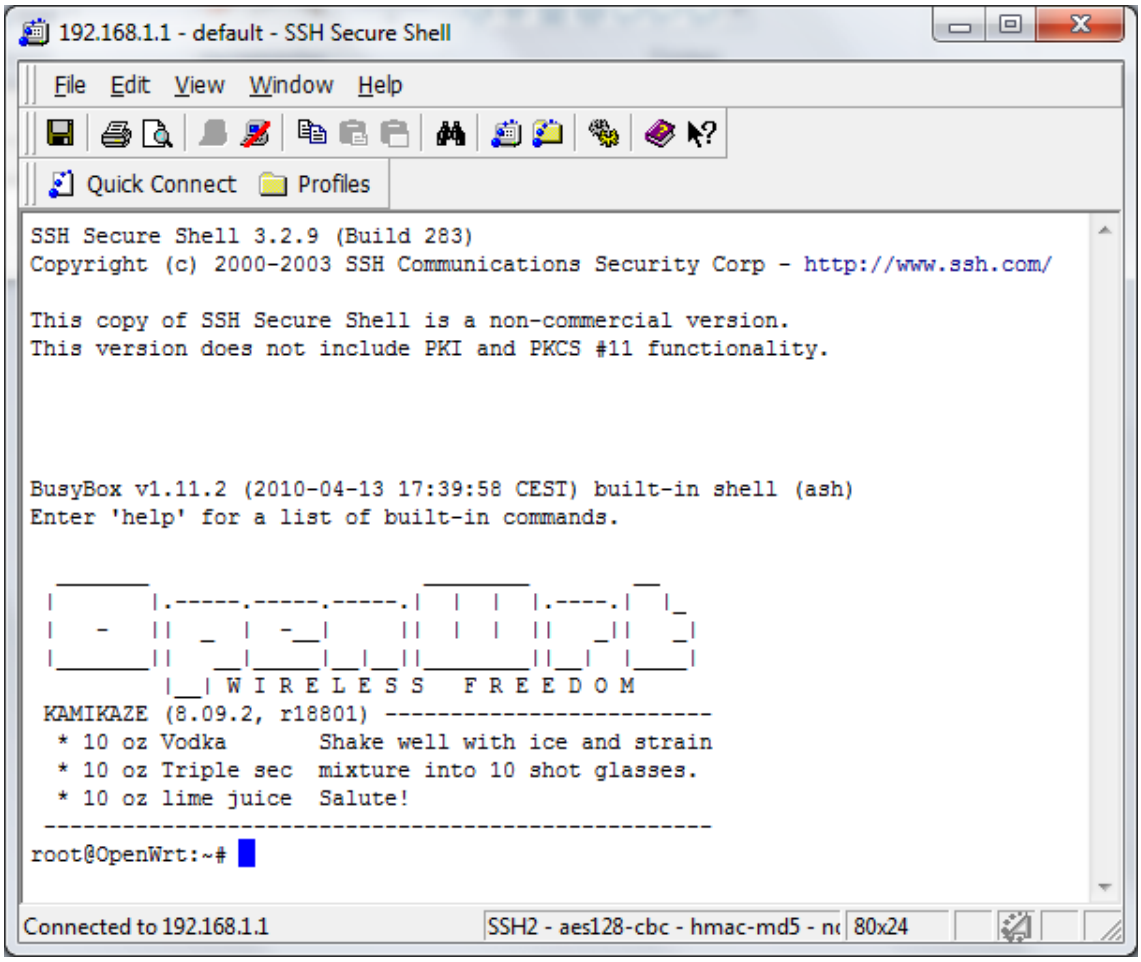



**Imagen 5. 14: Línea de comandos de OpenWrt**

a) <u>Accediendo mediante SSH desde Linux:</u>

1) Lo primero que haremos será abrir una consola de Linux (Terminal)

2) Ejecutaremos el siguiente comando:

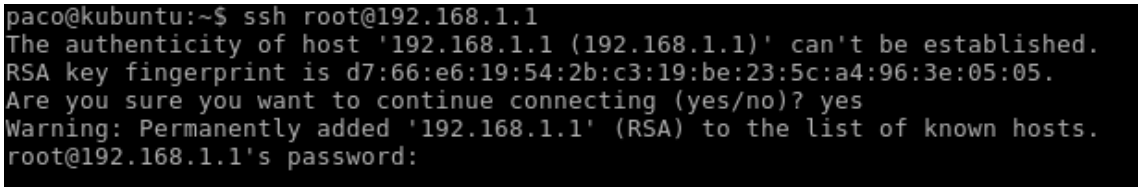



**Imagen 5. 15: SSH desde consola de Linux**

3) A continuación se nos pedirá el password

4) Por último ya hemos accedido a la línea de comandos de OpenWrt.

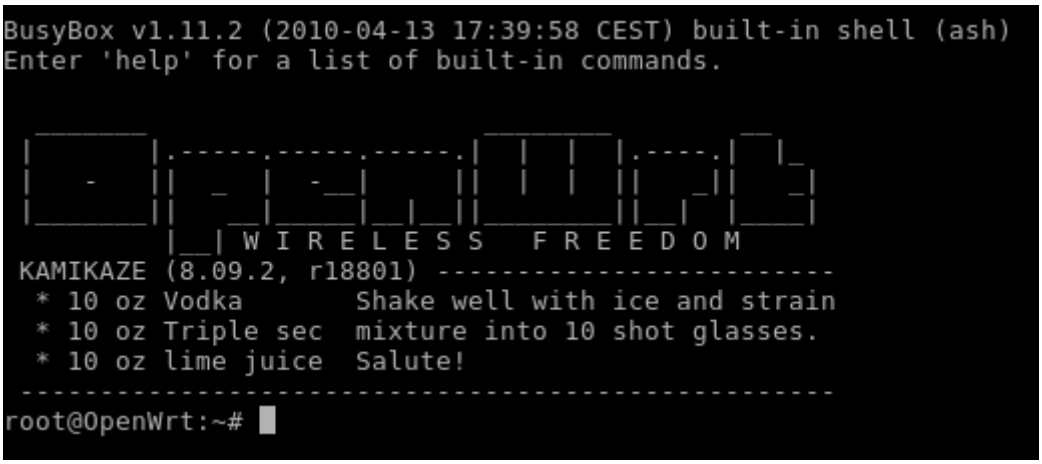



**Imagen 5. 16: Línea de comandos de OpenWrt**

### 5.5. Configurando los archivos network y wireless.

Desde la consola de SSH se pueden configurar los archivos network y wireless utilizando el comando "uci" o con un editor de texto de consola como "vi", en esta sección se mostrara como realizar la configuración desde el editor de texto de consola. Los comandos y las acciones a realizar son idénticas en Windows y en Linux.

Habrá que configurar el router para separar las redes ethernet y wireless que están encapsuladas en una interfaz br0 tal y como se muestra en la figura:

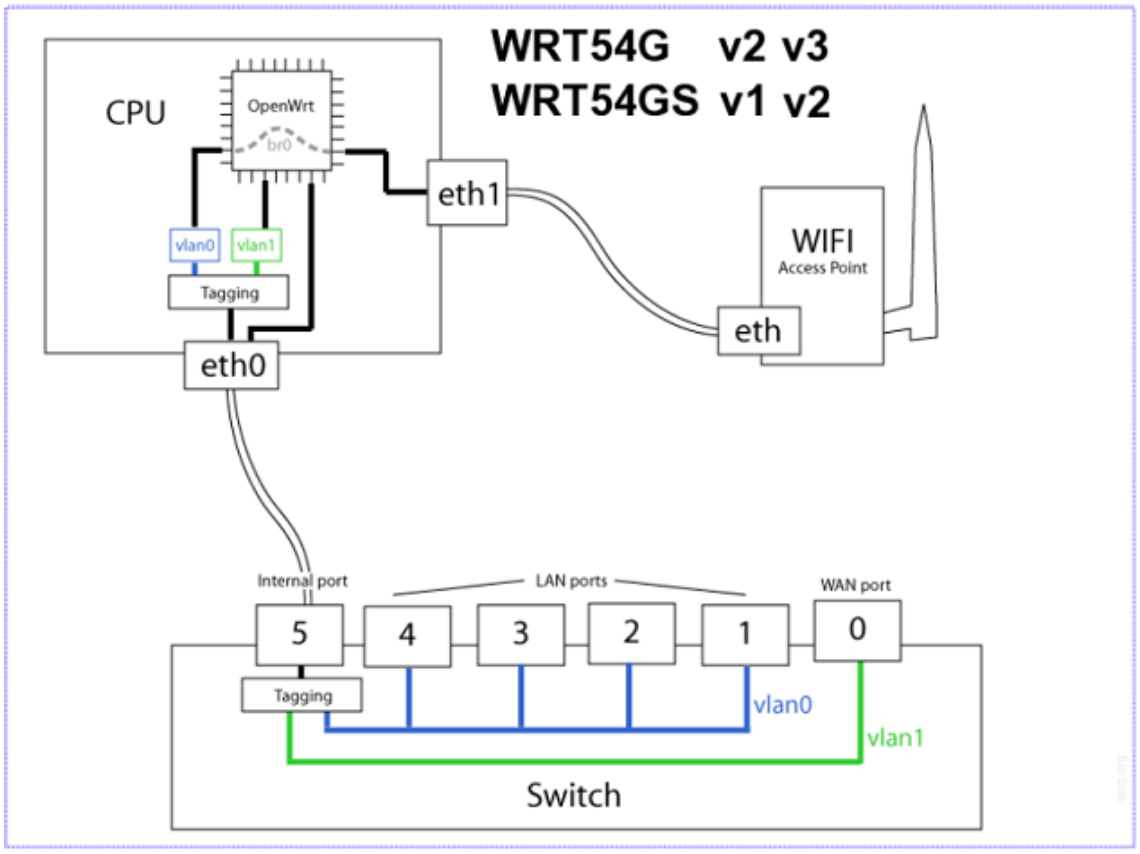



**Imagen 5. 17: Configuración de la red**

La razón por la que separaremos la red wireless de la red ethernet es para que las tramas de W2LAN no vayan a parar a los equipos conectados por cable al router.

Para editar la configuración del router, tendremos que movernos a la ruta /etc/config/ y procedemos a editar los archivos para dejarlos como se muestra a continuación:

- Archivo network.

*config 'switch' 'eth0'*

*option 'vlan0' '0 1 2 3 5*'*

*option 'vlan1' '4 5'*

*config 'interface' 'loopback'*

*option 'ifname' 'lo'*

*option 'proto' 'static'*

*option 'ipaddr' '127.0.0.1'*

*option 'netmask' '255.0.0.0'*

*config 'interface' 'lan'*

*option 'type' 'bridge'*

*option 'ifname' 'eth0.0'*

*option 'proto' 'static'*

*option 'ipaddr' '192.168.1.1'*

*option 'netmask' '255.255.255.0'*

*config 'interface' 'wan'*

*option 'ifname' 'eth0.1'*

*option 'proto' 'dhcp'*

*config 'interface' 'wifi'*

*option type bridge*

*option ifname "wl0"*

*option proto static*

*option ipaddr '192.168.5.1'*

*option netmask 255.255.255.0*

- Archivo wireless:

*config wifi-device wl0*

*option type broadcom*

*option channel 5*

*# REMOVE THIS LINE TO ENABLE WIFI:*

*option disabled 0*

*config wifi-iface*

*option device wl0*

*option network wifi*

*option mode adhoc*

*option ssid OpenWrt*

*option encryption none*

Con esta configuración, tenemos separado el switch del punto de acceso wifi en dos VLAN, de esta forma las tramas no interferirán en el funcionamiento del switch.

El protocolo W2LAN necesita que el punto de acceso este en modo adhoc, por eso se ha configurado de esta forma. En la red wireless el punto de acceso tiene la dirección IP 192.168.5.1.

Para poder conectarnos a la red inalámbrica, deberemos configurar el adaptador wifi de nuestro ordenador en modo Ad-hoc, con una dirección IP de la red 192.168.5.0/24, el gateway será la dirección del router (192.168.5.1) y el SSID será OpenWrt.

### 5.6. Creando el archivo binario de W2LAN (Compilación cruzada)

Una vez configurado el router y el adaptador inalámbrico de nuestro ordenador, ya estamos en disposición de crear el archivo ejecutable de W2LAN para comprobar que este funciona correctamente en el router, para más tarde poder crear un paquete capaz de instalarlo en el router con el gestor de paquetes opkg.

Vamos a explicar cómo realizar la compilación cruzada, supondremos que la descarga de la versión de desarrollo de OpenWrt se ha hecho en el directorio de nuestro usuario en Linux *(cd ~/):*

1) Lo primero que tenemos que hacer es exportar unas variables:

   *PATH=$PATH:"8.09/staging_dir/toolchain-mipsel_gcc3.4.6/bin"*

2) Lo siguiente será indicar el compilador de C que vamos a utilizar:

   *CC=mipsel-linux-uclibc-gcc*

3) A continuación debemos copiar unas librerías de libpcap de la versión de desarrollo de OpenWrt, esto es necesario debido a que cuando realizamos el configure, este no es capaz de encontrar la librería de libpcap:

   *cp –r 8.09/build_dir/mipsel/libpcap-0.9.8/ipkg-install/usr/lib 8.09/staging_dir/toolchain-mipsel_gcc3.4.6/*

   *cp -r 8.09/build_dir/mipsel/libpcap-0.9.8/ipkg-install/usr/include 8.09/staging_dir/toolchain-mipsel_gcc3.4.6/*

4) Una vez copiadas las librerías de libpcap, procederemos a descargarnos y descomprimir los archivos fuente de W2LAN:

   *wget http://labit501.upct.es/~fburrull/investigacion/w2lan/w2lan-0.1.tar.gz*

   *tar zxf w2lan-0.1.tar.gz*


**Imagen 5. 18: Directorios y archivos necesarios**

5) Una vez descomprimidos los archivos, ingresaremos en el directorio w2lan-0.1:

*cd w2lan-0.1/*

6) Ahora realizamos el configure marcándole que vamos a utilizar un compilador cruzado:

*./configure --host=mipsel-linux*

```
paco@kubuntu:~/PFC/w2lan-0.1$ ./configure --host=mipsel-linux
configure: WARNING: If you wanted to set the --build type, don't use --host.
    If a cross compiler is detected then cross compile mode will be used.
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for a thread-safe mkdir -p... /bin/mkdir -p
checking for gawk... gawk
checking whether make sets $(MAKE)... yes
checking for mipsel-linux-strip... no
checking for strip... strip
checking for mipsel-linux-gcc... no
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
checking whether gcc accepts -g... yes
checking for gcc option to accept ISO C89... none needed
checking for style of include used by make... GNU
checking dependency style of gcc... gcc3
checking for pcap_sendpacket in -lpcap... yes
checking for pcap_open_live in -lpcap... yes
checking for pcap_datalink_val_to_name in -lpcap... yes
checking for pcap_get_selectable_fd in -lpcap... yes
checking how to run the C preprocessor... gcc -E
checking for grep that handles long lines and -e... /bin/grep
checking for egrep... /bin/grep -E
checking for ANSI C header files... yes
checking for sys/types.h... yes
checking for sys/stat.h... yes
checking for stdlib.h... yes
checking for string.h... yes
checking for memory.h... yes
```

**Imagen 5. 19: Configure**

7) Una vez haya terminado el configure sin que nos haya dado errores, se nos habrán creado dos archivos Makefile, uno en el directorio actual y otro dentro del directorio src, los cuales deberemos modificar para eliminar la opción de depuración de la memoria dinámica (-Dmalloc=rpl_malloc), ya que el compilador cruzado no es capaz de compilar esta opción.

*cp Makefile Makefile.old*

*grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile*

*cd src*

*cp Makefile Makefile.old*

*grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile*

*cd ..*

8) Una vez eliminada la opción, ya podemos compilar el protocolo W2LAN realizando un make:

*make*

9) Cuando el make finalice, ya tendremos nuestro archivo binario dentro del directorio src listo para ser copiado al router y comprobar su funcionamiento:

*cd src*

*scp w2lan root@192.168.1.1:/tmp/*

10) Una vez copiado el archivo al router, accederemos a la línea de comandos de OpenWrt mediante SSH como se explicó anteriormente:

*ssh root@192.168.1.1*

11) Una vez hemos ingresado mediante SSH, tendremos que desplazarnos al directorio tmp que es donde se encuentra nuestro ejecutable:

*cd /tmp*

12) Comprobamos que el programa se ejecuta correctamente:

*./w2lan wl0 ó ./w2lan –h*

```
root@OpenWrt:/tmp# ./w2lan -h

w2lan - Layer 2 routing protocol for Ethernet wireless ad-hoc networks
Author: Francesc Burrull i Mestres <francesc.burrull@upct.es>

usage: ./w2lan [options] [device]

options:
      -m [mode]       valid modes: passive, reactive, always (see 'man 8 w2lan')
      -h              this help message and exit
      -v              display the version number and exit
device:
      [device]        wireless device where w2lan operates (i.e. eth2)
root@OpenWrt:/tmp#
```

**Imagen 5. 20: Ejecución de W2LAN en OpenWrt**

13) Una vez comprobado que funciona correctamente moveremos el archivo al directorio bin, para que seamos capaces de ejecutarlo desde cualquier ubicación:

*mv w2lan /bin*

**Nota:** si desea conservar el archivo ejecutable en el router, es necesario que lo mueva o copie a otra ubicación distinta al directorio *tmp*, ya que este es un directorio temporal y se borrará por completo en cada reinicio del router.

### 5.7. Paquete W2LAN (Makefile)

Una vez que se ha comprobado que OpenWrt es capaz de ejecutar el firmware, estamos en disposición de crear el paquete para poder instalar w2lan en el router.

Para crear paquetes, lo que necesitamos es escribir un archivo Makefile, con todos los datos de nuestro paquete.

El Makefile necesario para compilar W2LAN es el siguiente:

```
include $(TOPDIR)/rules.mk

PKG_NAME:=w2lan
PKG_RELEASE:=0.1

PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)-$(PKG_RELEASE)
PKG_SOURCE:=$(PKG_NAME)-$(PKG_RELEASE).tar.gz
PKG_SOURCE_URL:=http://labit501.upct.es/~fburrull/investigacion/w2l
an

PKG_BUILD_DEPENDS:=libpcap

include $(INCLUDE_DIR)/package.mk

define Package/w2lan
      SECTION:=w2lan
      CATEGORY:=w2lan
      DEPENDS:=+libpcap
      TITLE:=A protocol that transforms a wireless ad-hoc Ethernet
network into a LAN
endef

define Package/w2lan/description
      w2lan is a protocol that provides layer 2 routing for
wireless Ad-Hoc Ethernet networks.
      The effect is the elimination of the partial visibility
problem that MANET networks exhibit.

      Part of the parameter check consist on making sure that w2lan
is running on a 802.11 wireless
      Ad-Hoc Ethernet interface. Although the protocol can operate
in any Ethernet network, it is only
      in Ad-Hoc networks where the partial visibility problem
shows, hence w2lan proves useful (maybe
      in futures versions it can be an option for
testing/developing purposes, since not every PC has
      a wireless adapter, and most have Ethernet -or at least a
loopback-).

      w2lan splits each Ethernet frame into 3 new Ethernet frames:
w2lan announce (34 bytes), w2lan
```

```
      request (20 bytes) and w2lan data (the same size than the
original Ethernet frame).

      A node executing w2lan, reacts when a reception of a w2lan
frame occurs:

      If a w2lan announce frame is received, the node checks if
such conversation has been handled before.
      In the case it is a new conversation, the node updates its
conversations list and it sends the
      corresponding w2lan request frame.

      If a w2lan request frame is received, the node checks if he
has announced such conversation. If that
      is the case, a request counter is updated accordingly.

      If a w2lan data frame is received, the node checks if he has
requested such conversation. If that is
      the case, the original Ethernet frame is reconstructed with
the recently received w2lan data frame and
      the previously received w2lan announce frame. Next, the newly
reconstructed Ethernet frame is announced,
      and if the Ethernet frame destination field is either for the
node, multicast or broadcast, the frame is
      injected to the network (to be used by the node protocol
stack). After a certain time, specified in "w2lan.h"
      by TIMEOUT_REQUEST_SECONDS, the node will check the counter
associated with the newly announced frame, and
      if there is any interest (counter different of 0) the
corresponding w2lan data frame will be generated.

      Also, a node executing w2lan in "always" mode or sometimes in
"reactive" mode (see the OPTIONS section),
      when it sends an Ethernet frame, it also sends a
corresponding w2lan announce frame, that consist on a w2lan
      announce header plus a payload consisting on a new
conversation ID plus the header of the previously sent
      Ethernet frame. In a certain time, specified in "w2lan.h" by
TIMEOUT_REQUEST_SECONDS, the node might receive
      w2lan requests frames from its neighbours. If that is the
case, the node proceeds by sending a w2lan data frame,
      consisting on a w2lan data header, which includes the
conversation ID field, plus the payload of the previously
      sent Ethernet frame.
endef

define Build/Configure
      (cp -r "../../build_dir/mipsel/libpcap-0.9.8/ipkg-
install/usr/lib" "../../staging_dir/toolchain-mipsel_gcc3.4.6/"; \
      cp -r "../../build_dir/mipsel/libpcap-0.9.8/ipkg-
install/usr/include" "../../staging_dir/toolchain-
mipsel_gcc3.4.6/"; \
      cd $(PKG_BUILD_DIR); \
       wget $(PKG_SOURCE_URL)/$(PKG_SOURCE); \
      tar zxf $(PKG_SOURCE); \
      cp -rf $(PKG_NAME)-$(PKG_RELEASE)/* $(PKG_BUILD_DIR); \
      rm -rf $(PKG_NAME)-$(PKG_RELEASE); \
```

```
	./configure --host=$(GNU_TARGET_NAME); \
	cp Makefile Makefile.old; \
	grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile; \
	cd src; \
	cp Makefile Makefile.old; \
	grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile; \
	rm Makefile.old; \
	cd ..; \
);
endef

define Build/Prepare
	mkdir -p $(PKG_BUILD_DIR)/
endef

define Package/w2lan/install
	$(INSTALL_DIR) $(1)/bin
	$(INSTALL_BIN) $(PKG_BUILD_DIR)/src/w2lan $(1)/bin
endef

$(eval $(call BuildPackage,w2lan))
```

A continuación se procederá a explicar las diferentes secciones de este Makefile:

- **PKG_NAME:=w2lan** → Indica el nombre del paquete.
- **PKG_RELEASE:=0.1** → Indica la versión del paquete.
- **PKG_BUILD_DIR := $(BUILD_DIR)/$(PKG_NAME)-$(PKG_RELEASE)** → El directorio que se va a utilizar durante la compilación.
- **PKG_SOURCE:=$(PKG_NAME)-$(PKG_RELEASE).tar.gz** → Indica como se llama el archivo donde están comprimidos los archivos fuente.
- **PKG_SOURCE_URL:=http://labit501.upct.es/~fburrull/investigacion/w2lan** → Indica la dirección web desde la que se pueden descargar los archivos fuente.
- **PKG_BUILD_DEPENDS:=libpcap** → Indica que necesitamos la librería libpcap para poder ejecutar el programa en OpenWrt.
- **define Package/w2lan** → En esta sección se indica donde va a estar ubicado el paquete de W2LAN en el menuconfig, en nuestro caso se ha creado una nueva categoría llamada "w2lan", en esta sección se indica que es necesaria la librería libpcap para poder compilar y crear el paquete. También hay una pequeña descripción del paquete a modo de titulo.
- **define Package/w2lan/description** → En esta sección realizaremos una descripción más extensa del paquete, en esta descripción añadiremos modos de funcionamiento, opciones y todo la información necesaria del paquete.
- **define Build/Configure** → En este apartado se realizan todas las acciones previas a la compilación, aquí podemos observar todos los pasos realizados en el apartado anterior de este proyecto, pero de forma automatizada.
- **define Build/Prepare** → En este apartado, se crea el directorio a utilizar para compilar el paquete.
- **define Package/w2lan/install** → Aquí indicamos donde se va a copiar el archivo binario de W2LAN una vez instalado en OpenWrt.

- **$(eval $(call BuildPackage,w2lan))** → Por ultimo hacemos una llamada al compilador, para que compile el código fuente y genere el paquete de W2LAN.

### 5.8. Seleccionando el nuevo paquete en menuconfig.

Una vez tengamos nuestro archivo Makefile completo, tendremos que guardarlo en la ruta 8.09/package/w2lan/ para poder seleccionarlo en el menuconfig para crear el paquete.

Una vez guardado el archivo en el directorio que se ha indicado anteriormente, procederemos a realizar los cambios de configuración necesarios en el menuconfig:

1) Ejecutamos el comando make menuconfig:

*Make menuconfig*

2) Nos aparecerá la ventana de menuconfig. Podemos comprobar que hay una sección nueva llamada w2lan.

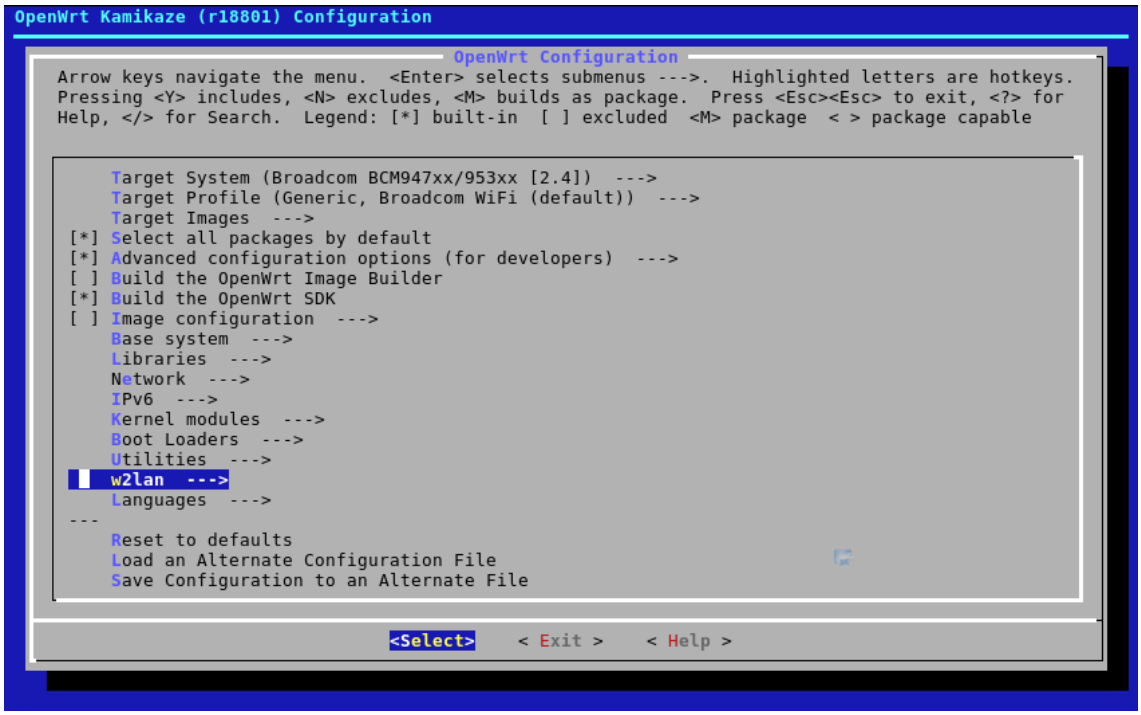



**Imagen 5. 21: Submenú W2LAN**

3) Accedemos a la nueva sección y vemos que podemos seleccionar w2lan para que se compile junto al firmware o para que se cree un nuevo paquete.

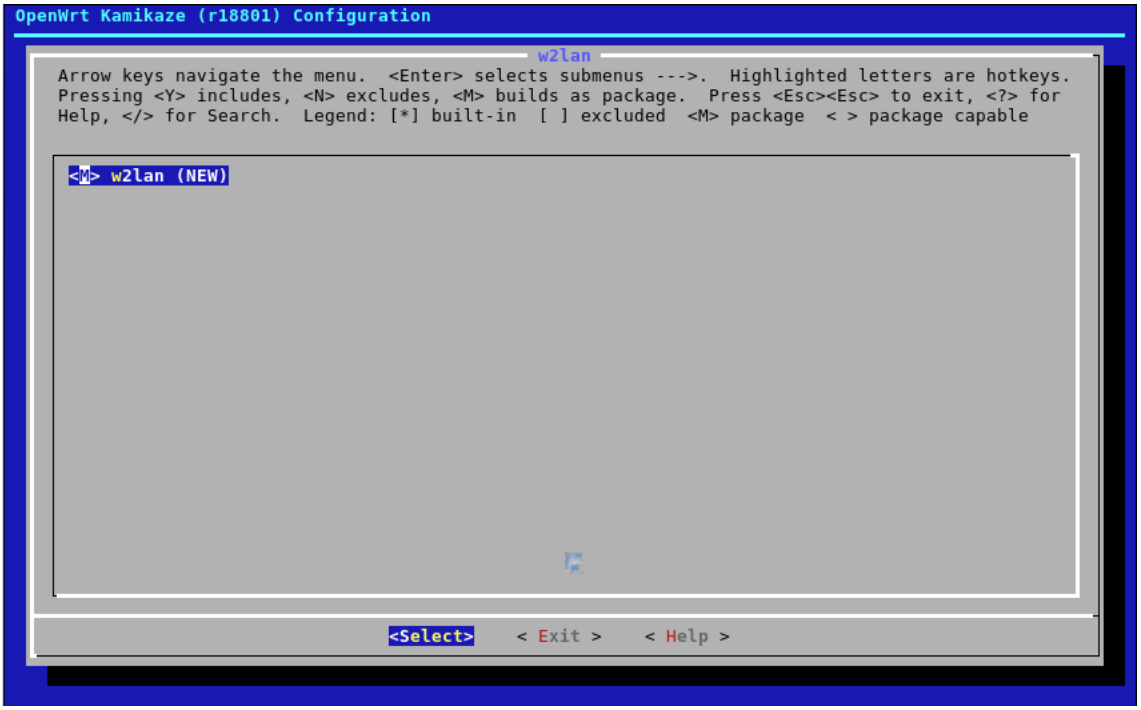



**Imagen 5. 22: Selección del paquete W2LAN**

## 5.9. Compilando el paquete.

A continuación procederemos a seleccionar lo que queremos hacer con el paquete:

- Seleccionamos la creación del paquete para realizar la instalación posteriormente (opción M):

  1) Ejecutamos:

     *Make package/w2lan/compile*

     También podemos añadir la opción V=99 si queremos que se nos muestre el proceso detallado de la compilación:

     *Make package/w2lan/compile V=99*

  2) Una vez se haya creado el paquete en este caso tendremos que copiar el paquete al router. El paquete se encuentra en el directorio 8.09/bin/packages/mipsel/

     *scp w2lan_0.1_mipsel.ipk root@192.168.1.1:/tmp/*

  3) Una vez copiado pasaremos al apartado 5.10 en el que se explica cómo instalar el paquete.

- Seleccionando la integración del paquete en el firmware (opción *): si optamos por esta opción lo único que debemos hacer es reinstalar y reconfigurar el

firmware como se ha explicado en capítulos anteriores. El nuevo firmware lo encontraremos en el interior del directorio *bin.*

### 5.10. Instalando el paquete.

Esto solo tiene sentido en caso que hayamos seleccionado la opción de crear el paquete y no lo hayamos incluido en el firmware.

Para poder instalar el paquete, tenemos que desplazarnos a la ubicación en la que hayamos copiado el paquete y ejecutaremos el comando:

*opkg install w2lan_0.1_mipsel.ipk*

Una vez instalado podemos comprobar si se ha instalado el paquete:

*opkg list_installed | grep w2lan*

### 5.11. Ejecutando W2LAN.

Para ejecutar w2lan lo único que tenemos que hacer es escribir el comando:

*w2lan [options] interface*

Con el router y el firmware utilizados en este proyecto de fin de carrera la interfaz que utilizaremos será wl0.

Las opciones disponibles en w2lan son

-m passive → El nodo funciona en modo pasivo.

-m reactive → El nodo funciona en modo reactivo (por defecto).

-m always →El nodo opera en modo siempre.

Para obtener ayuda sobre w2lan solo hay que utilizar la opción –h.

*w2lan -h*

**EJEMPLOS**

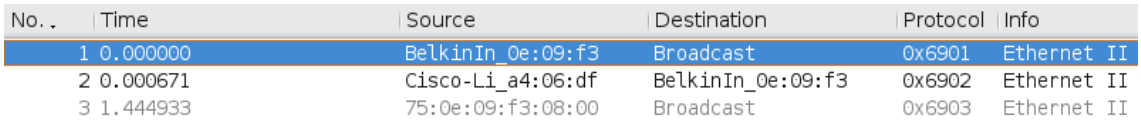
w2lan -m always eth2 → Este comando es recomendable cuando el nodo opera en modo servidor (el nodo es servidor de DHCP + NAT).

w2lan -m reactive eth2 → Este comando es recomendable cuando el nodo opera en modo cliente (el nodo accede a Internet mediante NAT).

w2lan -m passive eth2 → Este comando se recomendable cuando el nodo esta de ayudante (el nodo no tiene ningún usuario conectado, pero ayuda a aumentar el área de cobertura).

### 5.12. Captura de ping w2lan con wireshark.

Finalmente mostraremos una captura del programa wireshark para que podamos observar las tramas de w2lan.



| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 1 | 0.000000 | BelkinIn_0e:09:f3 | Broadcast | 0x6901 | Ethernet II |
| 2 | 0.000671 | Cisco-Li_a4:06:df | BelkinIn_0e:09:f3 | 0x6902 | Ethernet II |
| 3 | 1.444933 | 75:0e:09:f3:08:00 | Broadcast | 0x6903 | Ethernet II |

**Imagen 5. 23: Conversación W2LAN**

*Announce*

| FF FF FF FF FF FF | MAC origen | 69 01 |
|---|---|---|

| Conv. ID | Eth.MAC destino | Eth.MAC origen | Eth.T |
|---|---|---|---|

**Imagen 5. 24: Formato trama announce (W2LAN)**

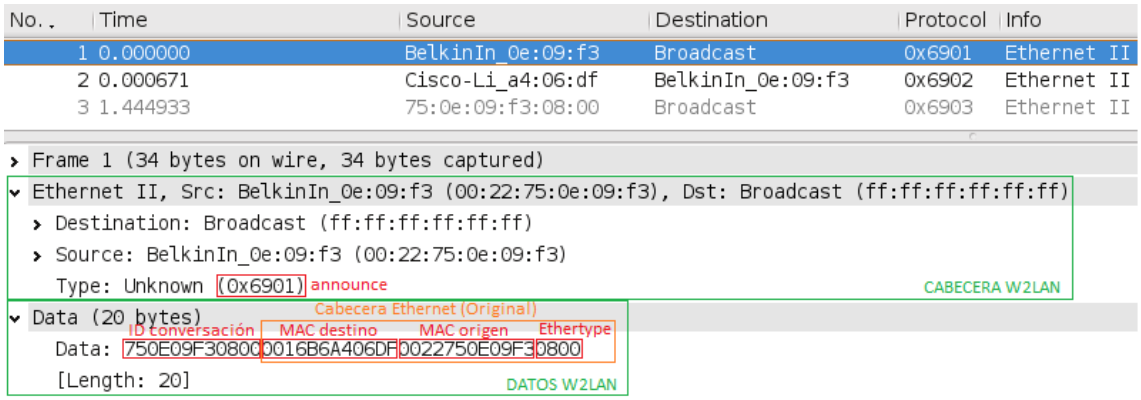



**Imagen 5. 25: Captura trama announce**

*Request*

| MAC destino | MAC origen | 69 02 |
|---|---|---|

| Conv. ID |
|---|

**Imagen 5. 26: Formato trama request (W2LAN)**

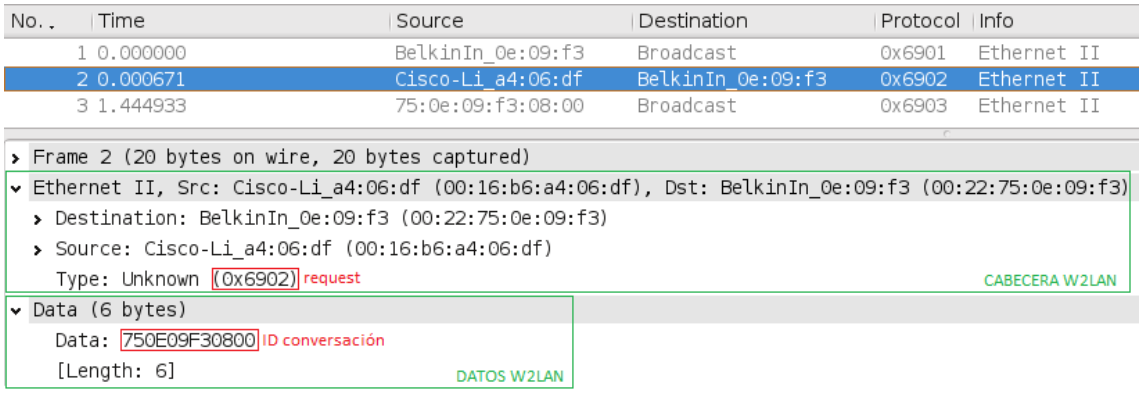



**Imagen 5. 27: captura trama request**

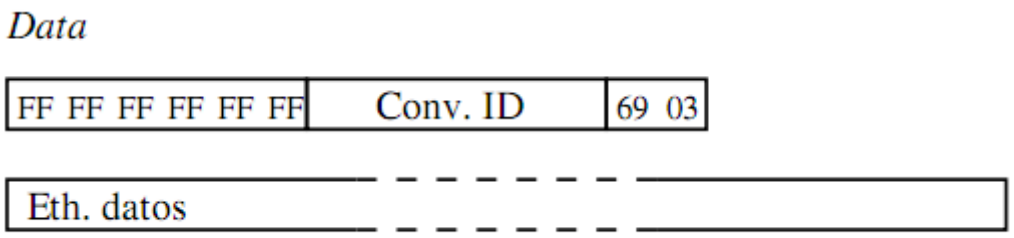



**Imagen 5. 28: Formato trama data (W2LAN)**

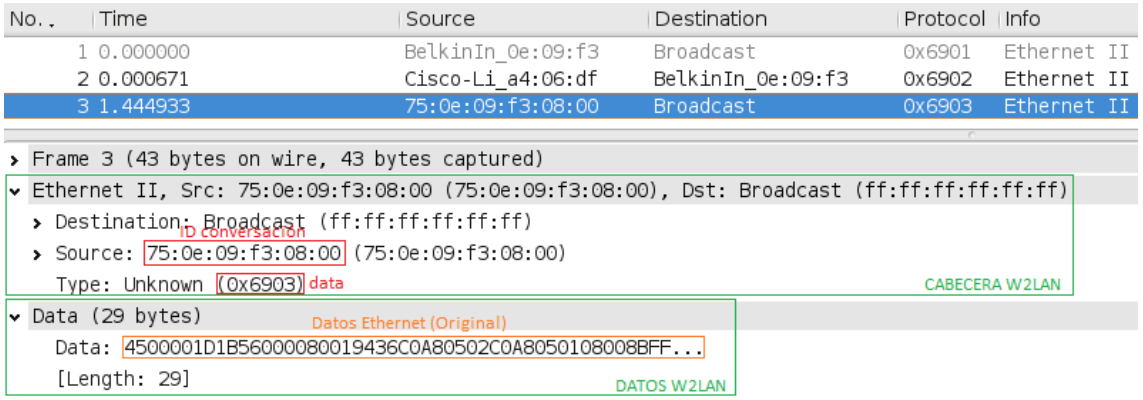



**Imagen 5. 29: captura trama data**

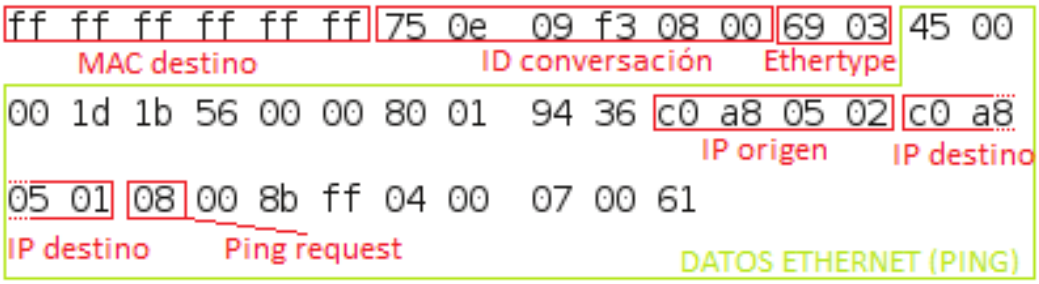



**Imagen 5. 30: Vista en bytes de la trama data (ping request)**

# 6. Líneas futuras.

Al comienzo del proyecto se nos presento el problema de ejecutar en un sistema empotrado un nuevo protocolo de comunicaciones.

De los posibles sistemas a los que se podía portar el protocolo, se selecciono la combinación de un firmware OpenWrt y un router Linksys wrt54gl.

Una vez realizada la selección de firmware, se realizaron las siguientes tareas para poder lograr el objetivo final:

- Seleccionar un router con punto de acceso WIFI: se han visto muchos modelos, pero en relación coste/calidad y posibilidad de instalar un sistema en el router de código abierto se eligió el router Linksys WRT54-GL.

- Herramientas y aplicaciones para implementar el sistema: sistema de desarrollo de OpenWrt.

- A partir de las selecciones iniciales se investigo sobre como compilar e instalar el protocolo W2LAN en el sistema final

Tras finalizar el proceso, se ha aprendido a configurar un router con un firmware basado en el kernel de Linux, a utilizar el sistema de desarrollo de OpenWrt y a utilizar un compilador cruzado.

Para poder seguir perfeccionando el proyecto, se recomienda:

- Implementar en kernel 2.6 de OpenWrt.

- Añadir opciones de configuración de W2LAN en la interfaz web del router.

- Implementar un paquete para el kernel.

- Añadir opciones de filtrado en iptables para filtrar tramas por ETHERTYPE.

- Portar a otros sistemas como puede ser videoconsolas (PSP, XBOX360, PS3, Wii…) y dispositivos móviles que dispongan de conectividad WI-FI (PDA, iPod, iPhone…)

# 7. Conclusiones.

Los objetivos de este proyecto de final de carrera se han completado con éxito obteniendo los siguientes resultados:

- Elección adecuada del sistema empotrado (router), debe ser un router que disponga de interfaz wireless y que soporte cambios de firmware. **Se ha elegido el router WRT54GL, router inalámbrico que puede soportar más de un sistema operativo.**

- Elección del firmware adecuado el cual soporte la instalación de paquetes. **Se ha seleccionado OpenWrt en su versión 8.09.**

- Compilar el protocolo W2LAN para que se pueda ejecutar en la arquitectura del router. **Se ha conseguido crear el archivo ejecutable de w2lan utilizando el compilador cruzado.**

- Crear un paquete con W2LAN para que se pueda instalar en cualquier router. **Se consiguió crear el paquete w2lan utilizando las herramientas de desarrollo de OpenWrt.**

- Comprobar que el router es capaz de ejecutar el protocolo W2LAN. **Se ha comprobado que el router es capaz de ejecutar W2LAN, se ha comprobado realizando la captura de una conversación con wireshark.**

# 8. Bibliografía


- Francesc Burrull i Mestres, "Contribución a la Evaluación y Diseño de Protocolos Broadcast para Redes LAN Ethernet y MANET", Tesis Doctoral, Junio 2005.

- Comunidad de desarrolladores de OpenWrt, http://www.openwrt.org/

- Manual de documentación de OpenWrt, http://wiki.openwrt.org/

# 9. Anexos.

## A) Anexo A: Breve descripción de los paquetes de OpenWrt.

- **base-files:** Estructura del sistema de archivos de OpenWrt y scripts.
- **base-files-brcm:** Arquitectura/Tabla de archivos específicos.
- **bridge:** Herramientas de puenteado Ethernet.
- **busybox:** Proporciona versiones reducidas de la mayoría de utilidades UNIX comunes en un pequeño archivo ejecutable.
- **dnsmasq:** Un servidor ligero de DNS y DHCP.
- **dropbear**: Un pequeño servidor/cliente SSH 2 diseñado para entornos de poca memoria.
- **haserl:** Un CGI envolvente para ejecutar archivos de shell script incrustados en documentos HTML.
- **opkg:** Un sistema de gestión de paquetes ligero.
- **iptables:** El programa cortafuegos de filtrado de red para Ipv4
- **iptables-mod-ipopt:** Extensiones de iptables (IPv4) para coincidencias/cambios en las opciones de paquetes IP.
- **iwlib:** Librería para ajustar las tarjetas WiFi usadas en la extensión wireless.
- **kernel-2.4.30-brcm:** Kernel de Linux (versión 2.4.30) adaptado para OpenWrt
- **kmod-brcm-wl:** Controlador propietario para dispositivos con chip de wireless Broadcom.
- **kmod-diag:** Módulos del kernel para los LED y los botones.
- **kmod-ipt-ipopt:** Módulos para el kernel de filtro de red (IPv4) para coincidencias/cambios en las opciones de paquetes IP.
- **kmod-ipt-nat-default:** Módulos predeterminados para el kernel de filtro de red (IPv4) para protocolos especiales.
- **kmod-ppp:** Soporte para PPP.
- **kmod-gre:** Soporte para el kernel de túneles GRE.
- **kmod-pppoe:** Soporte de PPP sobre Ethernet.
- **kmod-wlcompat**: Módulo de compatibilidad para usar la extensión wireless con chips wireless broadcom.
- **kmod-switch:** Controlador de switch para tipos de switch robo/admtek.
- **libopenssl:** Librerías OpenSSL (Secure Socket Layer).

- **libpcre:** Una librería compatible de expresiones regulares de Perl.
- **libsqlite2:** Motor (librería) de base de datos SQLite (v2.x).
- **lighttpd:** Un servidor web flexible y ligero.
- **lighttpd-mod-cgi:** Módulo CGI para lighttpd.
- **lighttpd-mod-redirect:** Módulo para redirecciones de URL.
- **mtd:** Herramienta para modificaciones el chip flash.
- **nvram:** Utilidad NVRAM y librerías para hardware de Broadcom.
- **php4-cgi:** Preprocesador de Hipertexto PHP4 (CGI).
- **php4-mod-sqlite**: Módulo SQLite para PHP4.
- **ppp:** Un demonio PPP (Protocolo Punto-a-Punto) con soporte MPPE/MPPC.
- **ppp-mod-pppoe:** Un módulo PPPoE (PPP sobre Ethernet) para PPP.
- **pptp:** Un cliente para el protocolo de túneles Punto-a-Punto (PPTP).
- **uclibc:** Librería estándar de C para sistemas Linux incrustados.
- **webif**: Una interfaz web modular y extensible para OpenWrt.
- **wificonf:** Utilidad de reemplazo para wlconf (configuración wireless).
- **wireless-tools:** Herramientas para el ajuste de tarjetas WiFi usando la extensión Wireless.
- **zlib:** Una implementación (librería) del algoritmo de compresión "deflate" (deflación).

## B) Anexo B: Helloworld en OpenWrt.

- Código de helloworld.

```
#include <stdio.h>

int main(void)
{
    printf("Hello world");
    return 0;
}
```

- Makefile:

```
include $(TOPDIR)/rules.mk

PKG_NAME:=helloworld

PKG_VERSION:=1

PKG_RELEASE:=1

PKG_BUILD_DIR:=$(BUILD_DIR)/$(PKG_NAME)-$(PKG_VERSION)

include $(INCLUDE_DIR)/package.mk

define Package/helloworld

  SECTION:=hello

  CATEGORY:=helloworld

  TITLE:= print a helloworld

  DESCRIPTION:=Print a helloworld

endef

define Build/Configure

  $(call Build/Configure/Default,--with-linux-headers=$(LINUX_DIR))

endef

define Package/helloworld/install

        $(INSTALL_DIR) $(1)/bin

        $(INSTALL_BIN) $(PKG_BUILD_DIR)/src/w2lan $(1)/bin

endef

$(eval $(call BuildPackage,helloworld))
```

## C) Anexo C: Script para compilación cruzada.

Para realizar la compilación cruzada de W2LAN, este script se debe ejecutar en el mismo directorio en el que se encuentra la carpeta 8.09, además debe estar descomprimido el archivo con las fuentes de W2LAN.

```
#!/bin/bash

DIR=$(pwd)"/8.09"

DIR2=$(pwd)

export PATH=$PATH:$DIR"/staging_dir/toolchain-mipsel_gcc3.4.6/bin"

export CC=mipsel-linux-uclibc-gcc

echo "PATH= "$PATH

echo "CC= "$CC

cp -r $DIR"/build_dir/mipsel/libpcap-0.9.8/ipkg-install/usr/lib"
$DIR"/staging_dir/toolchain-mipsel_gcc3.4.6/"

cp -r $DIR"/build_dir/mipsel/libpcap-0.9.8/ipkg-install/usr/include"
$DIR"/staging_dir/toolchain-mipsel_gcc3.4.6/"

echo "Entrando al directorio de W2LAN"

cd ./w2lan-0.1/

echo "Realizamos configure"

./configure --host=mipsel-linux

echo "Quitamos cadena xq no compila con ella"

cp Makefile Makefile.old

grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile

cd src

cp Makefile Makefile.old

grep -v " -Dmalloc=rpl_malloc" Makefile.old > Makefile

rm Makefile.old

cd ..

echo "compilamos el programa"

make

cp src/w2lan ../

echo "Regresando al directorio inicial"

cd $DIR2

echo "Ya puede copiar su ejecutable al router con el siguiente
comando"

echo "scp w2lan root@192.168.1.1:/tmp/"
```

## D) Anexo D: Tabla de prerrequisitos y sus correspondientes paquetes.

| Prerrequisito | Debian | Suse | Red Hat | OS X | Fedora |
|---|---|---|---|---|---|
| asciidoc | asciidoc | ? | ? | ? | ? |
| binutils | binutils | binutils | binutils | ? | binutils |
| bzip2 | bzip2 | bzip2 | bzip2 | ? | bzip2 |
| fastjar | fastjar | fastjar | ? | ? | — |
| flex | flex | flex | flex | ? | flex |
| g++ | g++ | gcc-c++ | ? | ? | gcc-c++ |
| gcc | gcc | gcc | gcc | ? | gcc |
| GNU autoconf | autoconf | autoconf | ? | ? | autoconf |
| GNU awk | gawk | gawk | gawk | ? | gawk |
| GNU bison | bison | bison | bison | ? | bison |
| gtk2.0-dev | libgtk2.0-dev | ? | ? | ? | gtk2-devel |
| intltool-update | intltool | intltool | ? | ? | — |
| jikes | jikes | jikes | ? | ? | — |
| libz-dev | zlib1g-dev | ? | zlib-devel | ? | zlib-devel |
| make | make | make | ? | ? | make |
| ncurses | ncurses-dev | ncurses-devel | ncurses-devel | ? | ncurses-devel |
| openssl/ssl.h | libssl-dev | libopenssl-devel | openssl-devel | ? | openssl-devel |
| patch | patch | patch | ? | ? | patch |
| perl-ExtUtils-MakeMaker | perl-modules | perl-ExtUtils-MakeMaker | perl-ExtUtils-MakeMaker | ? | perl-ExtUtils-MakeMaker |
| rsync | rsync | rsync | ? | ? | rsync |
| ruby | ruby | ruby | ? | ? | ruby |
| sdcc | sdcc | sdcc | ? | ? | sdcc |
| unzip | unzip | unzip | ? | ? | unzip |
| wget | wget | wget | wget | ? | wget |
| working-sdcc | sdcc | ? | ? | ? | — |
| xgettext | gettext | ? | ? | ? | gettext |

| | | | | | |
|---|---|---|---|---|---|
| xsltproc | xsltproc | libxslt | ? | ? | libxslt |
| zlib | zlib1g-dev | zlib-devel | zlib-devel | ? | zlib-devel |

Ejemplos de instalación:

- Debian 5.0:

  *# aptitude install autoconf bison flex gawk ncurses-dev unzip zlib1g-dev*

- Fedora 11:

  *# yum install autoconf binutils bison bzip2 flex gawk gcc gcc-c++ gettext make ncurses-devel patch unzip wget zlib-devel*

- openSuSE 11.1

  *# zypper install autoconf binutils bison bzip2 flex gawk gcc gcc-c++ gettext make ncurses-devel patch unzip wget zlib-devel*

- Ubuntu:

  *$ sudo apt-get install build-essential subversion libncurses5-dev zlib1g-dev gawk flex*