Índice general

Capítulo 1

Linux Embebido

Índice

1.1. Introducción

En este capitulo se muestra como se debe realizar para generar las herramientas de cross-compilación , sistema de archivos, imagen del kernel y la imagend del *Bootloader* para guardarlas en una memoria SD y así poder iniciar Linux Embebido en el IfLab.

Para la generación del las herramientas de cross-compilación y el sistema de archivos se uso *BuildRoot*. Para crear la imagen del kernel se descargo la versión 2.6.35 y para generar el u-boot se descargo el imx-bootlest-src versión 10.05.02.

1.2. Formato de la Tarjeta SD

Para poder crear una distribución de Linux Embebido para la IfLab es necesario dar un formato especial a una memoria SD, pues este sera el disco duro del sistema embebido.

Primero es necesario tener una memoria microSD de al menos 2GB de capacidad. Para darle el formato se deben seguir las siguientes instrucciones.

Primero se debe conectar la memoria al computador. Para mirar como es el nombre de esta memoria se ejecuta el comando *dmesg*. Para este caso el se tomara que el computador la detecto como */dev/sdb*.

Primero se debe desmontar la memoria, para esto se ejecuta el siguiente comando

```
[~]$ sudo umount /dev/sdb*
```

Luego con la aplicación fdisk se le dará formato a cada una de las particiones, para esto se ejecuta:

```
[~] $ sudo fdisk /dev/sdb
```

Esta aplicación obre una interfaz en la consola, el primer paso para realizar en fdisk es borrar las particiones existentes en la memoria. Esto se hace:

```
Command (m for help): d
Partition number (1-4):
```

En el campo de *Partition number* se debe ingresar todos los numero de las particiones hasta que no exista ninguna, para revisar eso basta poner el *Command* la letra *p*.

Luego de tener la tabla de particiones vacía se debe crear una primera partición de *200MB*. Para esto en fdisk se realiza:

```
Command (m for help): n               //Comando para crear una nueva particion
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p                 //Selecionar tipo de particion Primaria
Partition number (1-4, default 1): 1  //Nombrar la particion con el numero 1
```

Luego de esto el programa despliega desde donde es el inicio de la partición, para asignar el

valor por defecto solo hay que oprimir la tecla *Enter*. Luego se le debe indicar el final de la partición para esto se debe ingresar *+200M*. Estas instrucciones en la interfaz se ve como:

```
First sector (2048-7626751, default 2048):  //Presionar Enter
    Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-7626751, default 7626751):+200M
    Partition 1 of type Linux and of size 200 MiB is set
```

Ahora se debe crear la segunda partición en el espacio que queda en la microSD, para esto se realiza:

```
Command (m for help): n               //Comando para crear una nueva particion
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p                 //Selecionar tipo de particion Primaria
Partition number (1-4, default 1): 2  //Nombrar la particion con el numero 2
```

Luego se fija y el final de la partición por defecto:

```
First sector (411648-7626751, default 411648):
    Using default value 411648
Last sector, +sectors or +size{K,M,G} (411648-7626751, default 7626751):
    Using default value 7626751
Partition 2 of type Linux and of size 3.5 GiB is set
```

Ahora es necesario cambiar el tipo de partición 1 a OnTrack DM6 (53), en el programa se ejecuta:

```
Command (m for help):   t           // Comando para Cambiar tipo de Particion
Partition number (1-4): 1
Hex code (type L to list codes): 53
Changed system type of partition 1 to 53 (OnTrack DM6 Aux3)
```

Se realiza el mismo procedimiento para la partición 2 pero esta se cambia a Linux (83):

```
Command (m for help):   t           // Comando para Cambiar tipo de Particion
Partition number (1-4): 2
Hex code (type L to list codes): 83
Changed system type of partition 1 to 83 (Linux)
```

Al final de este proceso la tabla de particiones se debe ver de la siguiente manera:

```
Command (m for help):   p         // Comando para ver tabla
   Device Boot      Start         End      Blocks   Id  System
```

```
/dev/sdb1            2048      411647      204800   53  OnTrack DM6 Aux3
/dev/sdb2          411648     7626751     3607552   83  Linux
```

Para finalizar la creación de las particiones se deben guardar los cambios de la tabla de particiones, para esto se ejecuta:

```
Command (m for help): w        //Comando para Guardar la tabla
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Teniendo lista las particiones de la microSD es necesario dar formato a la segunda partición, para esto se ejecuta:

```
[~]$ sudo mkfs.ext3 /dev/sdb2
```

Teniendo listo el formato de la segunda partición la tarjeta microSD esta lista para almacenar el *bootlader*, la imagen del kernel y el sistema de archivos para la distribución de Linux Embebido.

1.3. Herramientas de Cross-Compilación y Sistema de Archivos

Buildroot es un conjunto de *Makefiles* y parches para la generación fácil de Linux embebido. Puede generar todas las herramientas de cross-compilación, el sistema de archivos, la imagen del kernel y la imagen del *Bootlader*. Esta herramienta tiene soporte para arquitecturas: ARM, MIPS, X86, PowerPC, entre otras. Sus características mas importantes son, la facilidad de configuración, debido las diferentes interfaces de configuración(*menuconfig*) que presenta y el soporte a cientos de paquetes para aplicaciones y librerías para ser usadas en espacio de usuario. [1]

1.3.1. Descarga y configuración

Para realizar la descarga de *Buildroot* se puede realizar siguiendo los pasos que se muestran la pagina oficial de esta proyecto[1], sin embargo para IfLab se encuentra disponible una versión disponible en el siguiente repositorio:

```
1  https://bitbucket.org/cicamargoba/mini-open/downloads/
```

Allí se descarga el archivo *buildroot-2013.08.1.tar.bz2*. Luego de descomprimir este archivo se debe ubicar desde la consola en esa carpeta. En conosola se ejecuta el siguiente comando:

```
1  [buildroot-2013.08.1]$ make menuconfig
```

Este comando abre la interfaz de configuración de *BuildRoot*, allí se pueden realizar todas tipo configuraciones para la creación de las herramientas de cross-compilación y los paquetes y librerias que se añadirán al sistema de archivos. En esta distribución de *BuildRoot* ya se le ha dado soporte a aplicaciones como : boa, hostapd, minicom, node, openocd, dhcp, entre otras. Cuando se ejecuta este comando en su consola se debe mostrar una interfaz de configuración como se muestra en la Figura **1-1**.

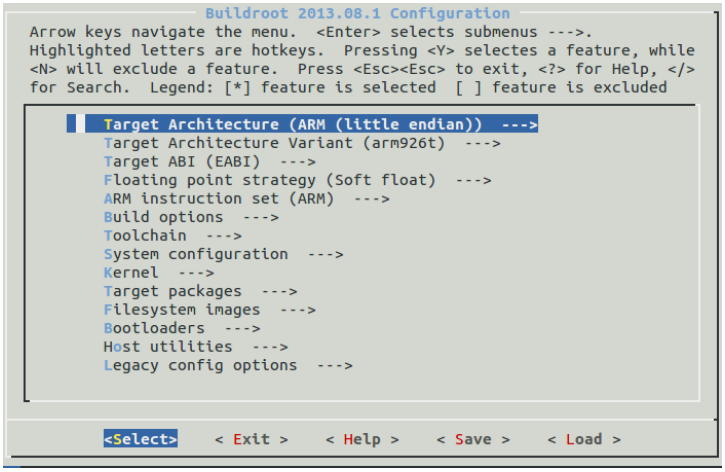

Figura **1-1**: Interfaz de configuración de Buildroot

En la configuración que trae por defecto ya se encuentran realizadas las configuraciones de la arquitectura y la generación de las herramientas de cross-compilación.

Es necesario ingresar a la opción de *System Configuration* para realizar la configuración del puerto donde se ubicara al consola, la velocidad de este puerto, además de configuraciones

[1] http://buildroot.uclibc.org/download.html

como el nombre del Usuario-Host y contraseña. A continuación se muestra como debe quedar las configuraciones del sistema.

```
System configuration  --->
     (buildroot) System hostname
     (Welcome to Buildroot) System banner
         Passwords encoding (md5)  --->
         /dev management (Static using device table)  --->
         Init system (Busybox)  --->
     (system/device_table.txt) Path to the permission tables
     (system/device_table_dev.txt) Path to the device tables
         Root FS skeleton (default target skeleton)  --->
     ()  Root password
     (ttyAM0) Port to run a getty (login prompt) on
         Baudrate to use (115200)  --->
     (vt100) Value to assign the TERM environment variable
     [*] remount root filesystem read-write during boot
     ()  Root filesystem overlay directories
     ()  Custom scripts to run before creating filesystem images
     ()  Custom scripts to run after creating filesystem images
```

En la opción *Port to run a getty* se especifica el puerto serial donde ubica la consola, por tanto es necesario cambiar de *ttyS0* a *ttyAM0*. Este cambio es necesario debido a que el kernel 2.6.35 apunta al puerto serial con ese nombre. Sin embargo si se usa una versión de kernel 3.7 se debe cambiar por *ttyAMA0*.

1.3.2. Módulos de nodeJs

Además de realizar configuraciones del sistema la interfaz de configuración de *BuilRoot* permite agregar o quitar aplicaciones y librerías del sistema de archivos. En este caso es necesario agregar módulos de nodeJs para realizar el desarrollo de la interfaz gráfica.

Para realizar esta configuración se debe ubicar en la opción de nodeJs.

```
 Target Package ->
     Interpreter languages and scripting  --->
         [*]nodejs
         Module Selection  --->
             [*] NPM for the target
             [*] Express web application framework
             [*] CoffeeScript
             (socket.io jade serialport) Additional modules
```

Los módulos adicionales que se necesitan para node se deben escribir en la opción de *Additional modules*. Para este caso se adicionaron los modulos: *socket.io, jade, serialport*.

Con estas modificaciones ya esta listo para realizar su compilación, esto se reliza ejecutando el siguiente comando en consola:

```
[buildroot-2013.08.1]$ make
```

Este proceso dura algunas horas, dependiendo de la conexión a internet y la capacidad de su computador de compilar todos los paquetes. Durante este proceso se presenta un error en la compilación de la aplicación openocd. Cuando esto ocurra se debe ejecutar en consola la siguiente instrucción:

```
[buildroot-2013.08.1]$ make CFLAGS=-std=gnu99
```

Luego si se presenta algún error solo basta ejecutar en consola de nuevo el siguiente comando:

```
[buildroot-2013.08.1]$ make
```

Cuando todo haya conculido en la consola se debe ver el siguiente mensaje:

```
[buildroot-2013.08.1]$ >>>   Generating root filesystem image rootfs.tar
```

Esto indica que todo el proceso de generación de herramientas y compilación de aplicaciones ha concluido. Este proceso crea un archivo llamado *rootfs.tar* con todo el sistema de archivos ubicado en *buildroot-2013.08/output/images*.

1.3.3. Herramientas de Cross-compilación

Las herramientas de Cross-Compilación se encuentran en el directorio: *buildroot-2013.08.1 /output/host/usr/bin/*.Es necesario crear un link al enlazador, esto se realiza con los siguientes comandos:

```
[buildroot-2013.08.1]$ cd output/host/usr/bin/
[bin]$ ln -s  arm-buildroot-linux-uclibcgnueabi-ld.real arm-linux-ld.real
```

Para usar estas herramientas de cross-compilación se debe agregar este directorio a la variable de entorno $\$PATH$. Esto se logra agregando la siguiente instrucción en el archivo *.bashrc*.

```
export PATH=$PATH:/path_buildroot/buildroot-2013.08.1/output/host/usr/bin
```

Los ejecutables del croos-compilador tienen el prefijo *arm-linux-*. Teniendo esto listo ya se pueden cross-compilar las aplicaciones.

1.3.4. Almacenamiento Sistema de Archivos

Teniendo listo el sistema de archivos generado por *BuildRoot*, solo queda guardar este en la memoria microSD, para esto se debe ir al directorio donde se encuentra el archivo *rootfs.tar*. Para descomprimir el sistema de archivos en la tarjeta se ejecuta:

```
1 [images] $ sudo  tar -jxvf rootfs.tar  -C /media/disk/
```

De esta forma ya se tiene almacenada el sistema de archivos en la memoria microSD.

1.4. Compilación del Kernel y del Imx-Bootlest

Con las herramientas de cross-compilación listas, se puede compilar la imagen del kernel de linux. Primero se deben descargar los archivos fuente de la versión de kernel a compilar, en este caso se uso el kernel 2.6.35, debido a que tiene soporte de audio para el procesador $IMX233$. La descarga se puede realizar del pagina web de desarrollo del kernel de Linux[2]. Sin embargo se puede descargar una versión de kernel ya configurado para esta arquitectura, esta versión se encuentra en el siguiente repositorio:

```
1 https://bitbucket.org/cicamargoba/mini-open/downloads/
```

El archivo se llama *linux-2.6.35.3.tar.bz2*, se debe descargar y descomprimir. Luego para revisar la configuración de cross-compilación se ejecuta en consola el siguiente comando:

```
1 [linux-2.6.35.3]$ make ARCH=arm CROSS_COMPILE=arm-linux- menuconfig
```

Se debe abrir en la consola un interfaz de configuración como la que se muestra en la Figura **1-2**

Allí se configura para que tipo de arquitectura se esta compilando, las opciones de arranque y cuales drivers se deben incluir en la compilación.

[2]https://www.kernel.org/

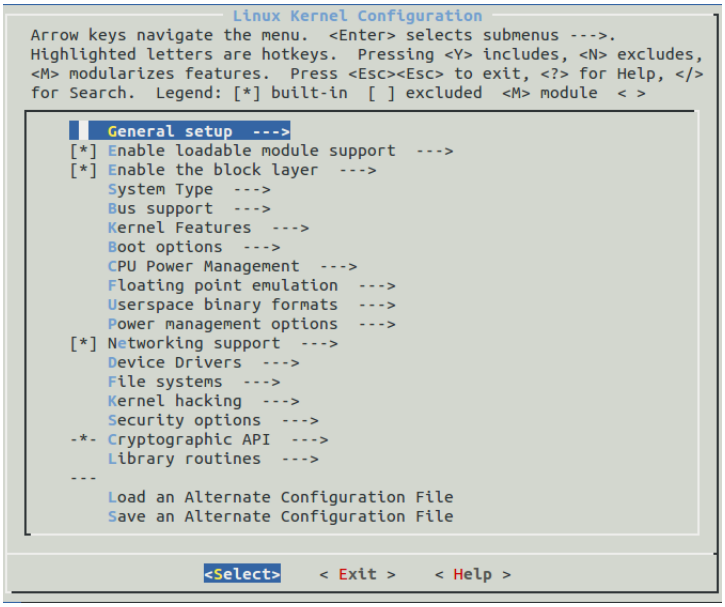

Figura **1-2**: Interfaz de configuración del Kernel 2.6.35

1.4.1. Driver de la Llave WiFi RTL8192CU

Para darle soporte a esta Llave Wifi debe seleccionar las siguientes opciones:

```
Device Drivers  --->
    [*] Networking support  --->
        [*]   Wireless LAN  --->
            <*>   Realtek 8192C USB WiFi
```

Esta llave WiFi, por defecta esta configurada para realizar metodos de ahorro de energia, sin embargo esto hace que a la hora de estar conectado se aumenten los tiempos de ejecución de la aplicación, para esto se debe deshabilitar esta opción de ahorro de energía. Para esto sse debe abrir el archivo Makefile del driver:

```
[linux-2.6.35.3]$ gedit drivers/net/wireless/rtl8192cu/Makefile
```

En este archivo se debe cambiar la opción de *CONFIG_POWER_SAVING*:

```
Cambiar : CONFIG_POWER_SAVING = y
por: CONFIG_POWER_SAVING = n
```

1.4.2. Compilación del Kernel 2.6.35

Con la configuración completa, es hora de realizar la compilación del kernel. Para esto se debe ejecutar en la consola el siguiente comando:

```
[linux-2.6.35.3]$ make ARCH=arm CROSS_COMPILE=arm-linux-
```

Dependiendo de las especificaciones de su computador el proceso tarda unos minutos. Cuando termine este proceso se debe observar el siguiente mensaje:

```
Kernel: arch/arm/boot/zImage is ready
```

En esta linea se informa donde esta ubicado el archivo con la imagen comprimida del Kernel.

1.4.3. Compilación del Imx-Bootlest

Teniendo lista la imagen del Kernel es necesario crear el *Boot loader*. Para esto es necesario descargar del repositorio el archivo *Imx-bootlets-src-10.05.02.tar.bz2*. Luego de descomprimir su contenido se ubica la consola en ese directorio.

Primero es necesario crear un link de *elftosb2*, para eso solo es necesario ejecutar la siguiente instrucción:

```
[Imx-bootlets-src-10.05.02] $ sudo ln -s `pwd`/elftosb2 /usr/sbin/
```

Ahora se puede realizar la compilación del *Boot-loader* y el almacenamiento en la primera partición de la microSD.Para esto se ejecuta:

```
[Imx-bootlets-src-10.05.02] $ make ARCH=mx23 CROSS_COMPILE=arm-linux-
```

Al finalizar este proceso se guardo el *Boot-loader* en la primera partición y la tarjeta microSD esta lista para conectar en IFLab.

Capítulo 2

Configuración HostPot

Índice

2.1. Introducción

2.2. Cross-Compilación Hostapd

Hostapd es un demonio en espacio de usuario para crear puntos de acceso y servidores de autenticación. Este programa implementa el estándar IEEE 802.11 para puntos de acceso, el IEEE 802.1X/WPA/WPA2/EAP de autenticación.[3]

Debido al uso de la Llave Wifi RTL8192CU se debe usar una versión de Hostapd especial que ofrece el fabricante de la llave, esta versión esta especialmente modificada para que funcione con esta llave wifi.

En el repositorio se encuentra un archivo llamado *Hostapd.tar* donde se encuentra esta versión especial lista para ser cross-compilada. Luego de realizar la descarga y descomprimir el archivo, se debe revisar el Makefile y debe indicar que se va realizar la compilación con el prefijo *arm-linux-gcc*.

Al finalizar este proceso se crea un archivo ejecutable llamado *hostapd*, este ejecutable se debe copiar en la carpeta */usr/sbin* del sistema de archivos del Embebido.

2.3. Configuración en el Sistema de archivos

2.3.1. Asignación de dirección IP

Teniendo la imagen del kernel y el sistema de archivos andando se debe verificar que nombre usa el sistema para identificar la llave WiFi. Para esto se ejecuta:

```
[~]# iwconfig
 wlan0   IEEE 802.11b  ESSID:"IfLab"  Nickname:"<WIFI@REALTEK>"
     Mode:Master  Frequency:2.412 GHz  Access Point: 44:33:4C:61:C3:18
     Bit Rate:11 Mb/s   Sensitivity:0/0
     Retry:off   RTS thr:off   Fragment thr:off
     Encryption key:off
     Power Management:off
     Link Quality:0  Signal level:0  Noise level:0
     Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
     Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

Ahora es necesario subir la la interfaz WiFi y asignarle una dirección IP, para esto se ejecuta en la consola:

```
[~]# ifconfig wlan0 192.168.50.1 up
```

2.3.2. Configuración Hostapd

Teniendo la la aplicación *hostapd* lista y la interfaz de WiFi configurada, se deben crear los archivos de configuración y así crear el punto de acceso en el IfLab.

Primero se debe crear el archivo de configuración del *hostapd*. Primero se crea una carpeta llamada *hostapd* en la carpeta etc del sistema de archivos.

```
[etc]# mkdir hostapd
```

En esta carpeta se crea un archivo llamado *hostapd.conf* el cual contiene la siguiente información:

```
[etc]# more hostapd.conf
    interface=wlan0
 driver=rtl871xdrv
 ssid=IfLab
 channel=1
 bridge=0
```

En este archivo se especifica la interfaz wifi que se va usar. Se le debe especificar cual driver debe usar para implementar el punto de acceso para esta llave WiFi es *rtl871xdrv*. En este archivo también se define el nombre de la red y el canal donde se va ubicar la red. Para correr la aplicación con esta configuración se ejecuta:

```
[~]# hostapd /etc/hostapd/hostapd.conf &
```

Al terminar la ejecución ya esta creado el demonio del *hostpot*, ya la red esta creado, sin embargo la conexión se debe realizar manual pues aún no se ha configurado el servidor *DHCP*, para la asignación automática de *IP*.

2.3.3. Configuración Servidor DHCP

En una red de comunicaciones es necesario asignar direciones IP a la hora de establecer una conexión, *ISC-DHCP* es una aplicación que ofrece la asignación de estas direcciones de forma fácil y rápida. Este es un programa .ºpen source"que implementa un protocolo de configuración dinámica en el *host* para conectar a un una red IP.[2]

Primero se debe crear un archivo donde indique cual de las interfaces de red usara por defecto. Para esto en el sistema embebido se ejecuta:

```
[etc]# mkdir default  //Crear carpeta
[default]# more isc-dhcp-server //Muestra el contenido del archivo
     INTERFACES="wlan0"
```

Luego en la carpeta */etc/dhcp* se debe crear un archivo llamado *dhcp.conf*. Este archivo contiene:

```
 [dhcp] # more dhcpd.conf
 subnet 192.168.50.0 netmask 255.255.255.0 {
 range 192.168.50.5 192.168.50.20;
 option domain-name-servers 192.168.50.1, 192.168.50.1;option routers 192.168.50.1;
 }
```

Teniendo la configuración lista para correr el servidor *DHCP* se debe ejecutar:

```
[~] # touch /var/lib/dhcp/dhcpd.leases &
[~] # dhcpd -cf /etc/dhcp/dhcpd.conf wlan0 &
```

En este momento ya esta corriendo el servidor *DHCP* y la conexión a la red inalámbrica se realiza de forma automática.

2.3.4. Inicio automático

En este momento ya esta configurado tanto el punto de acceso a la red como el servidor de direcciones IP, solo queda configurar el sistema para que inicie automáticamente y no se tenga que ejecutar de nuevo las instrucciones descritas anteriormente. Para esto se debe crear un nuevo archivo en */etc/init.d*.

El archivo debe tener:

```
[~] # more /etc/init.d/S51hostapd
ifconfig wlan0 192.168.50.1 &
hostapd /etc/hostapd/hostapd.conf &
touch /var/lib/dhcp/dhcpd.leases &
dhcpd -cf /etc/dhcp/dhcpd.conf wlan0 &
```

Ademas se le deben dar permiso de ejecución a este archivo, para esto se ejecuta:

```
[~] # chmod +x /etc/init.d/S51hostapd
```

De esta forma queda configurada el punto de acceso en el IfLab.

Bibliografía

[1] BuildRoot. *BuildRoot About*. 2014. URL: http://buildroot.uclibc.org/about.html (visitado 28-05-2014).

[2] Internet Systems Consortium. *ISC-DHCP About*. 2014. URL: http://www.isc.org/downloads/dhcp/ (visitado 28-05-2014).

[3] Hostapd. *Hostapd About*. 2014. URL: http://hostap.epitest.fi/hostapd/ (visitado 28-05-2014).