

# Compilacion Cruzada

---

## Introduccion

---

La compilacion de codigo fuente realizado bajo una determinada arquitectura que genera codigo ejecutable para una arquitectura diferente se llama **compilacion cruzada** .

En la compilacion cruzada hay dos tipos de sistemas

**huesped** : donde nos brinda la oportunidad de aprovechar los recursos que disponemos

**objetivo** : donde se ejecuta el codigo.

para realizar este tipo de compilacion nos apoyamos de **Builroot**.

## Builroot

---

Esta es una herramienta que simplifica y automatiza el proceso de construccion de un sistema embebido linux completo, usando compilacion cruzada.

## Requerimientos de sistemas

Builroot esta diseñado para correr en sistema linux.

Esta es una lista general de los paquetes obligatorios para ejecutar *builroot*.

- which
- sed

- make (version 3.81 or any later)
- binutils
- build-essential (only for Debian based systems)
- gcc (version 2.95 or any later)
- g++ (version 2.95 or any later)
- bash
- patch
- gzip
- bzip2
- perl (version 5.8.7 or any later)
- tar
- cpio
- python (version 2.6 or any later)
- unzip
- rsync
- wget +git

## Compilacion para la Raspberry Pi 2

---

### Sistema huesped

A continuacion se muestra una descripcion del CPU que sera utilizado como *huesped*

- Architecture: x86\_64
- CPU op-mode(s): 32-bit, 64-bit
- Byte Order: Little Endian
- CPU(s): 12
- On-line CPU(s) list: 0-11
- Thread(s) per core: 2

- Core(s) per socket: 6
- Socket(s): 1
- NUMA node(s): 1
- Vendor ID: GenuineIntel
- CPU family: 6
- Model: 45
- Model name: Intel(R) Xeon(R) CPU E5-2620 0 @ + 2.00GHz
- Stepping: 7
- CPU MHz: 1814.062
- BogoMIPS: 4000.22
- Virtualization: VT-x
- L1d cache: 32K
- L1i cache: 32K
- L2 cache: 256K
- L3 cache: 15360K
- NUMA node0 CPU(s): 0-11

## Obtencion de buildroot

Primero tenemos que obtener directamente de la pagina de buildroot

```
$ wget https://buildroot.org/downloads/buildroot-2016.02.tar.gz
```

## Descomprimir el archivo

Una vez que tenemos el archivo bastara con descomprimirllo

```
$ tar xvzf buildroot-2016.02.tar.gz
```

# Configuracion para Raspberry Pi 2

Nuestro sistema objetivo esta descrito a continuacion.

- Architecture: armv7l
- Byte Order: Little Endian
- CPU(s): 4
- On-line CPU(s) list: 0-3
- Thread(s) per core: 1
- Core(s) per socket: 4
- Socket(s): 1

Accedemos al directorio descomprimido y como la Raspberry Pi 2 es una tarjeta popular ya existe un archivo de configuracion paa esta tarjeta ( como tambien lo existe para la version 1 y recientemente para la version 3)

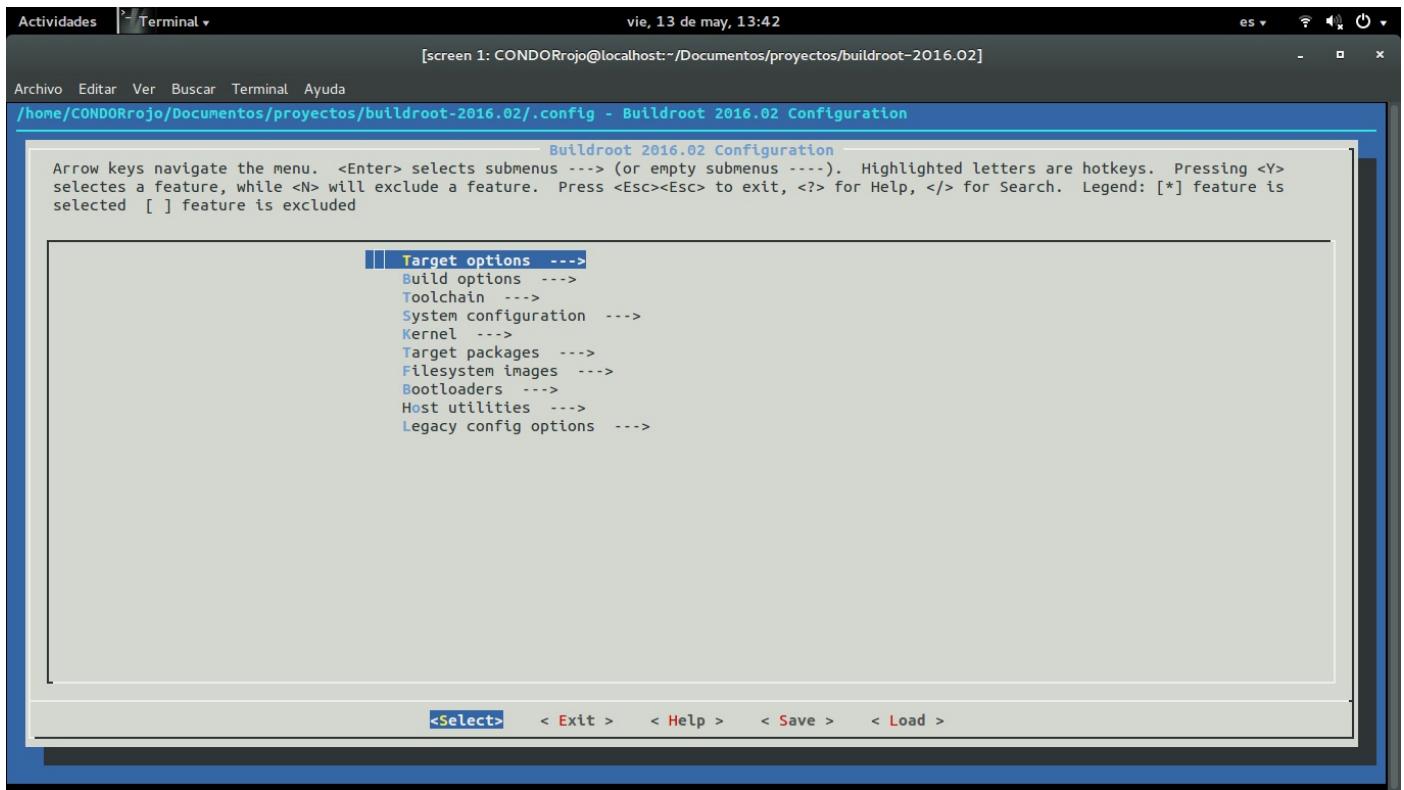
Ahora simplemente hay que ejecutar el siguiente comando:

```
$ make raspberrypi2_defconfig
```

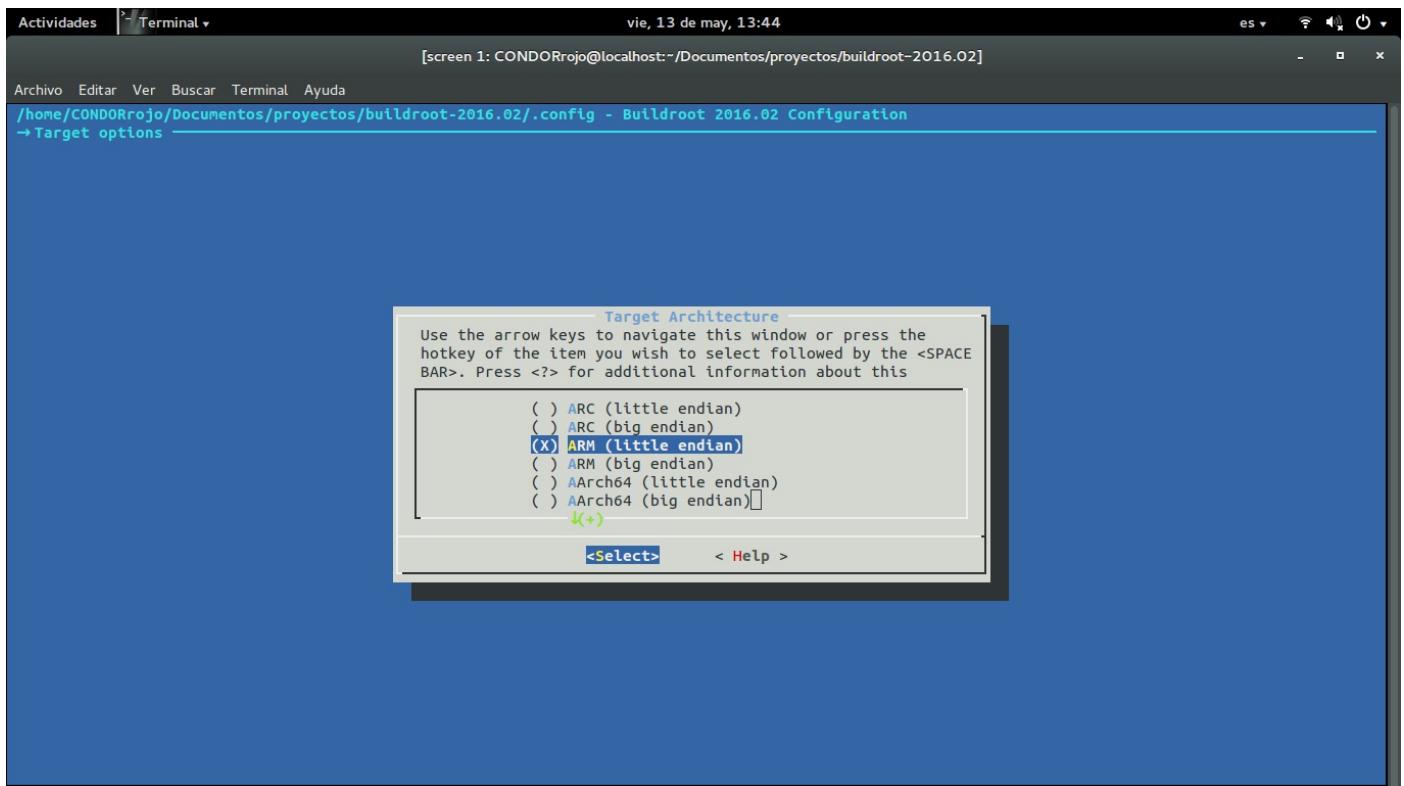
Ahora para agregar paquetes a nuestra compilacion simplemnte ejecutamos:

```
$ make menuconfig
```

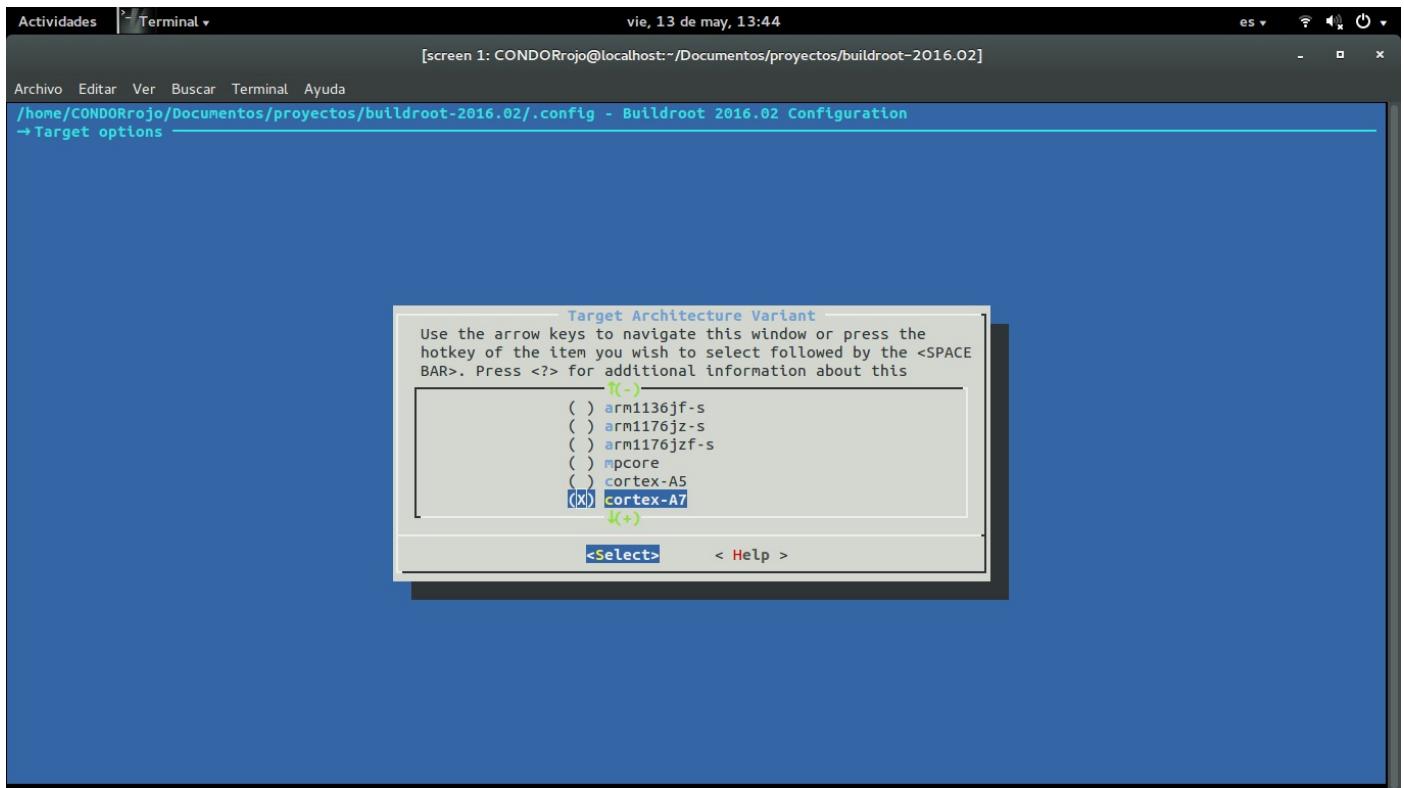
Nos muestra ventanas donde podemos elegir paquetes o opciones de compilacion que previamente no incluia el archivo "*raspberrypi2\_defconfig*" y podemos ver de manera grafica las opciones cargadas en el archivo **config**.



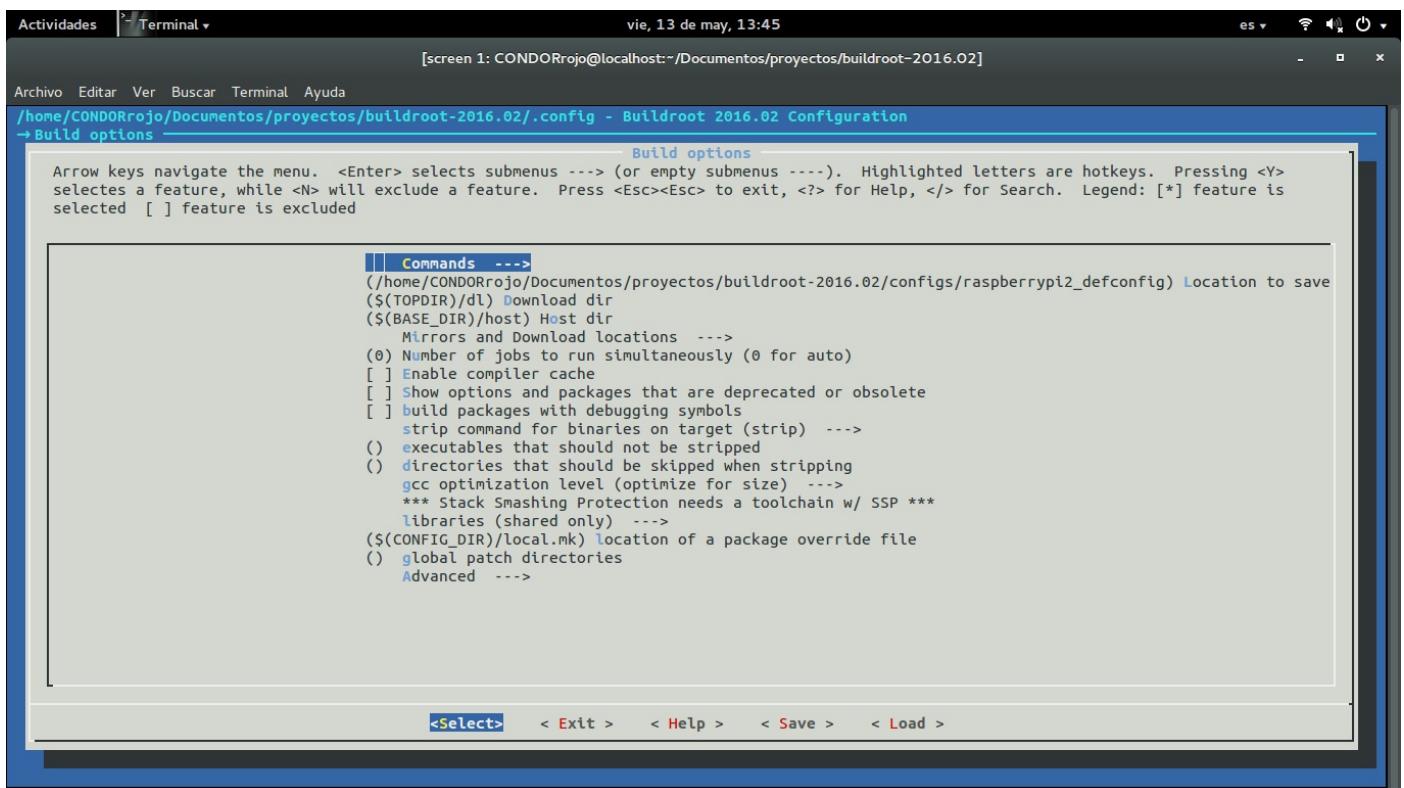
Entrando a la opcion de "Target Architecture" podemos notar que la arquitectura "ARM (little endian)" ya esta seleccionada debido al "*raspberrypi2\_defconfig*"



Lo mismo sucede para la "Variante"

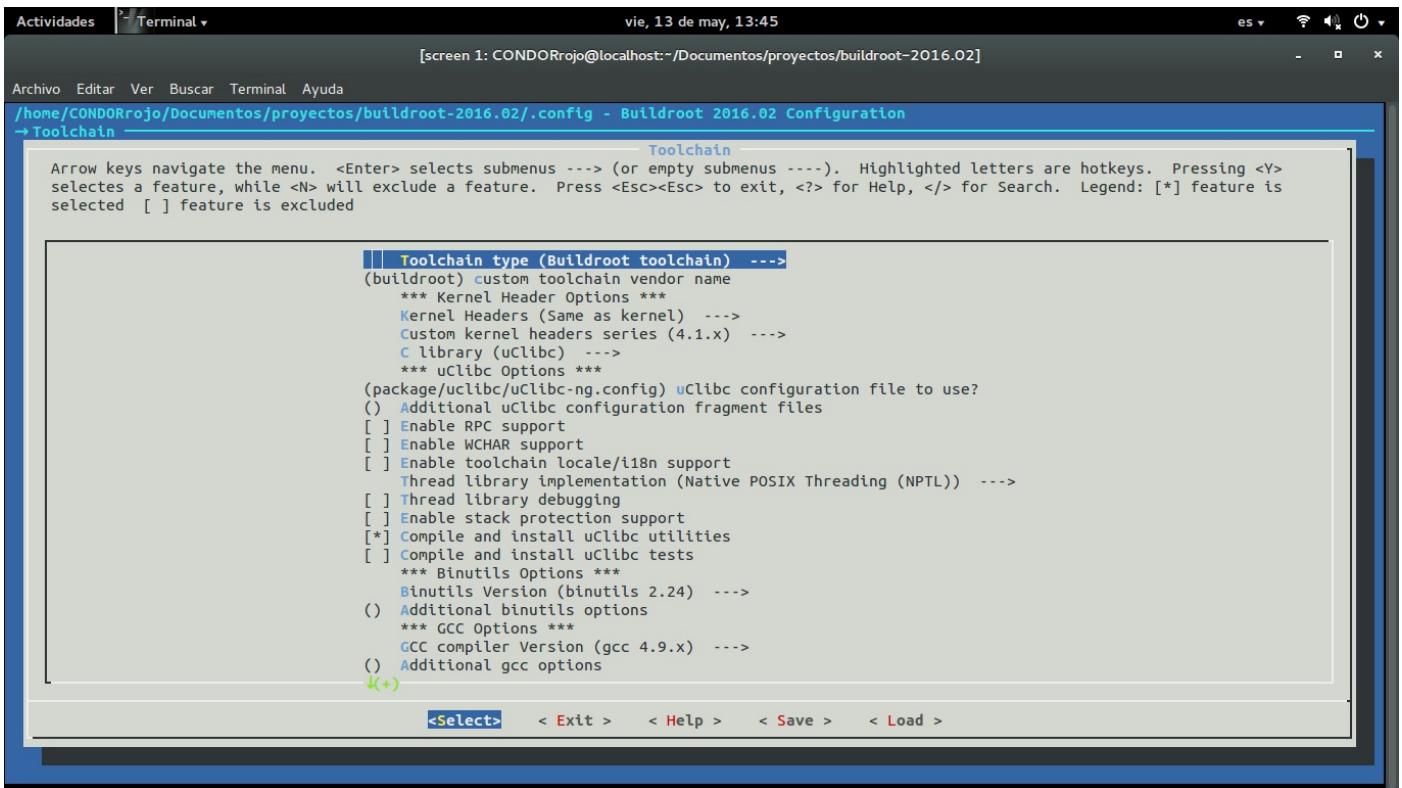


Tambien tenemos las opciones de construir, donde se indica los directorios de descarga y host. Tambien el numero de trabajos que corren simultanemente y varias opciones que se pueden manipular para optimizar la construccion.

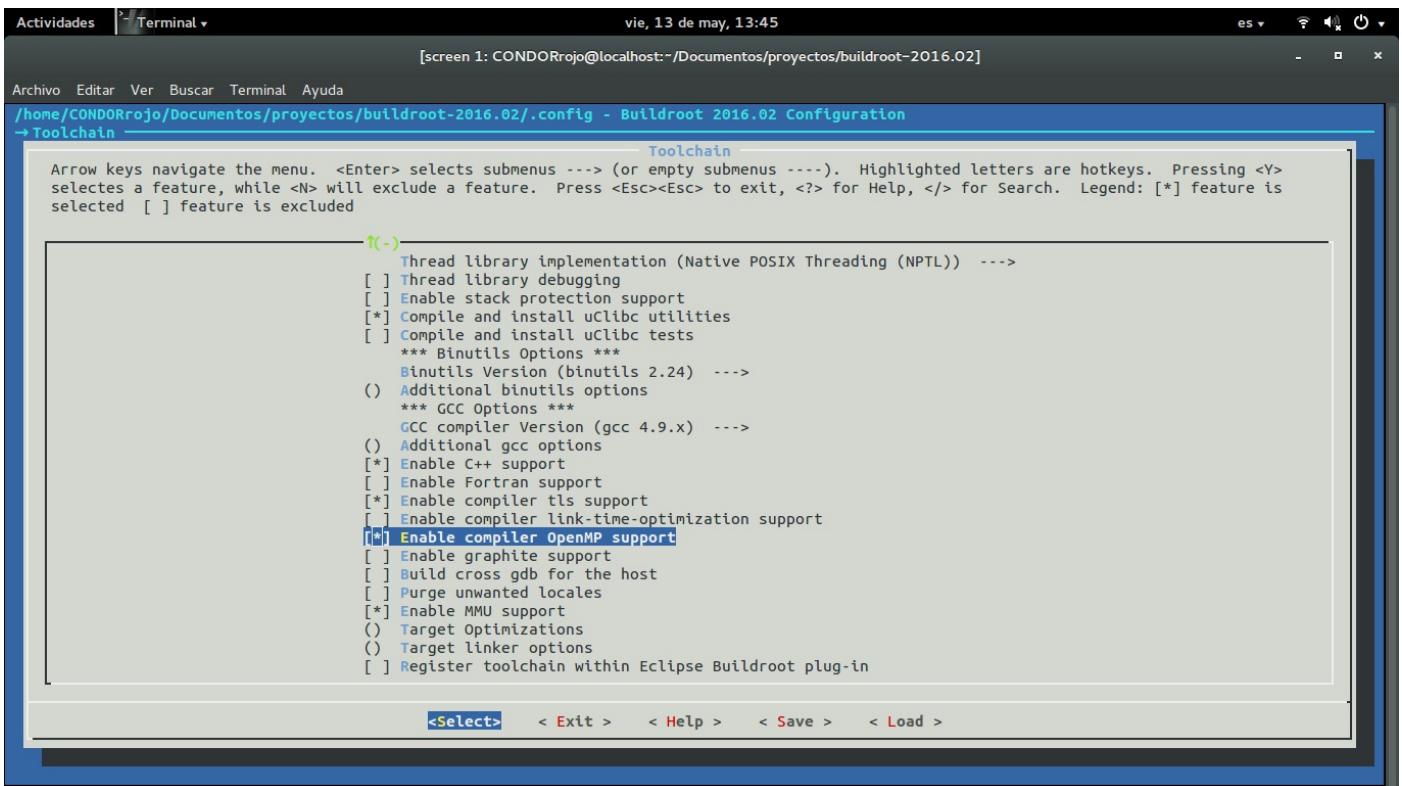


Lo mismo ocurre para el "Toolchain", donde podemos indicar opciones para cabeceras del kernel (versiones), opciones de GCC como seria la

versión que queremos instalar en nuestro sistema.

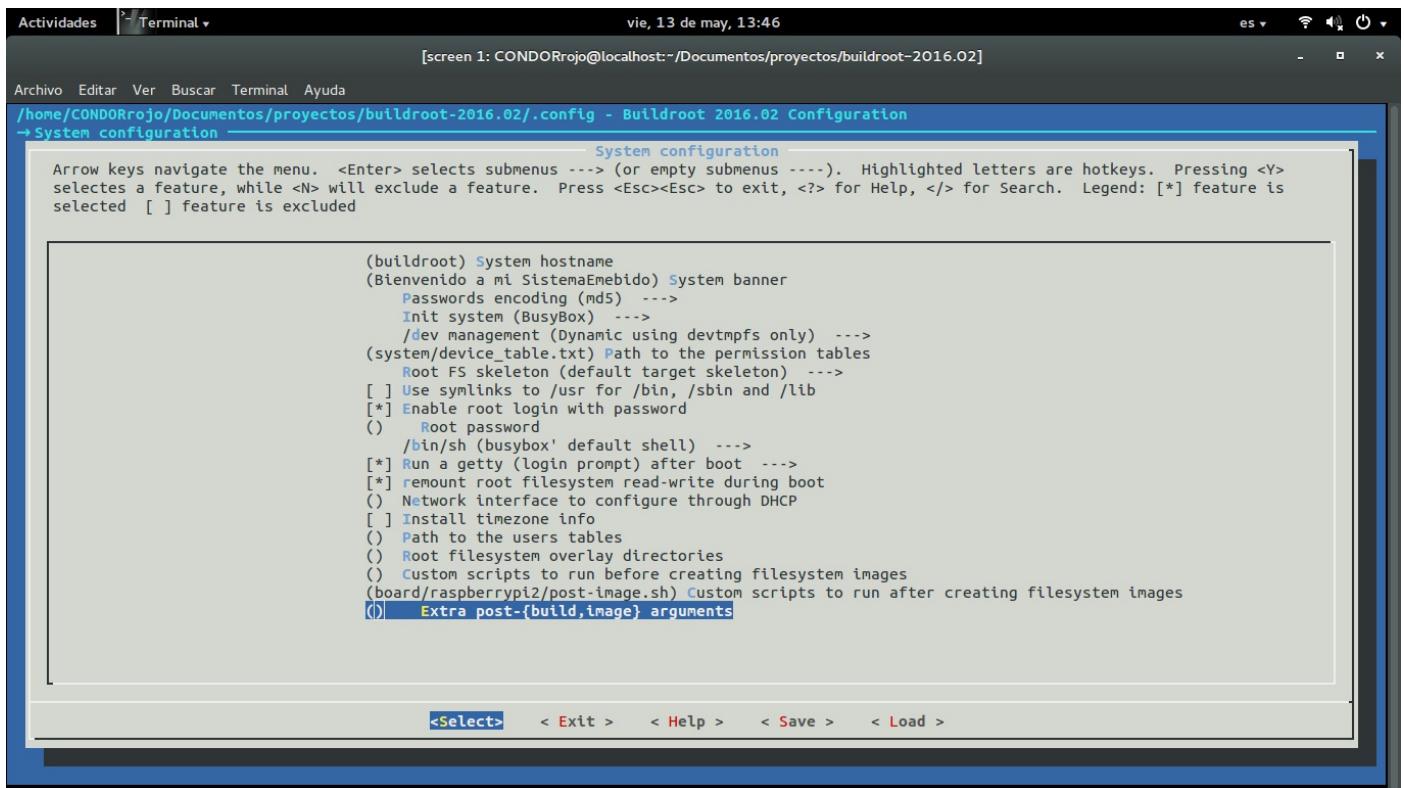


Así como Soporte para otro lenguajes como C++ o compilar OpenMP (Que en mi caso activo ya que quiero trabajar con procesamiento paralelo)

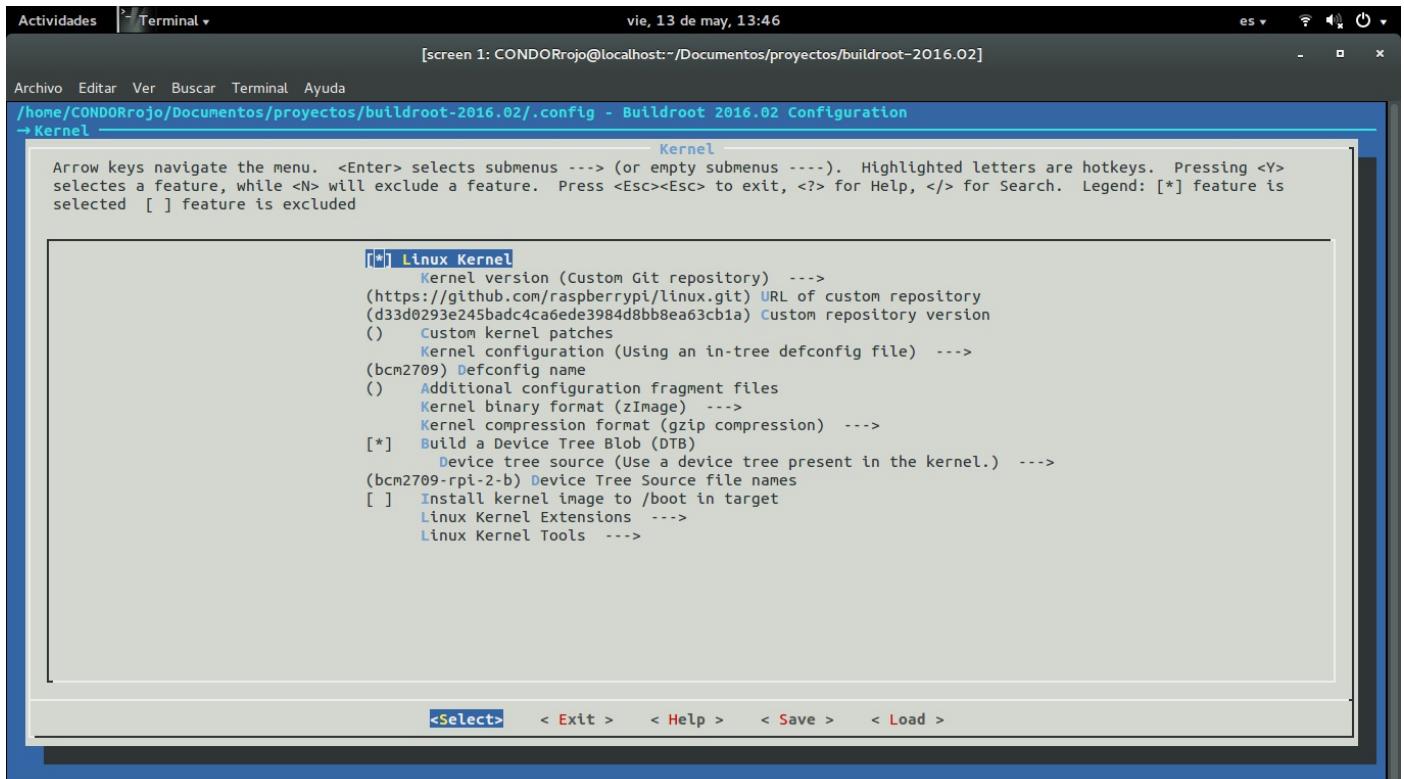


Otra opción es "System Configuration", donde entre algunas de las opciones que se especifican son el "System banner", El tipo de codificación

de contraseñas, especificar si el login del root es con contraseña e indicar la misma.

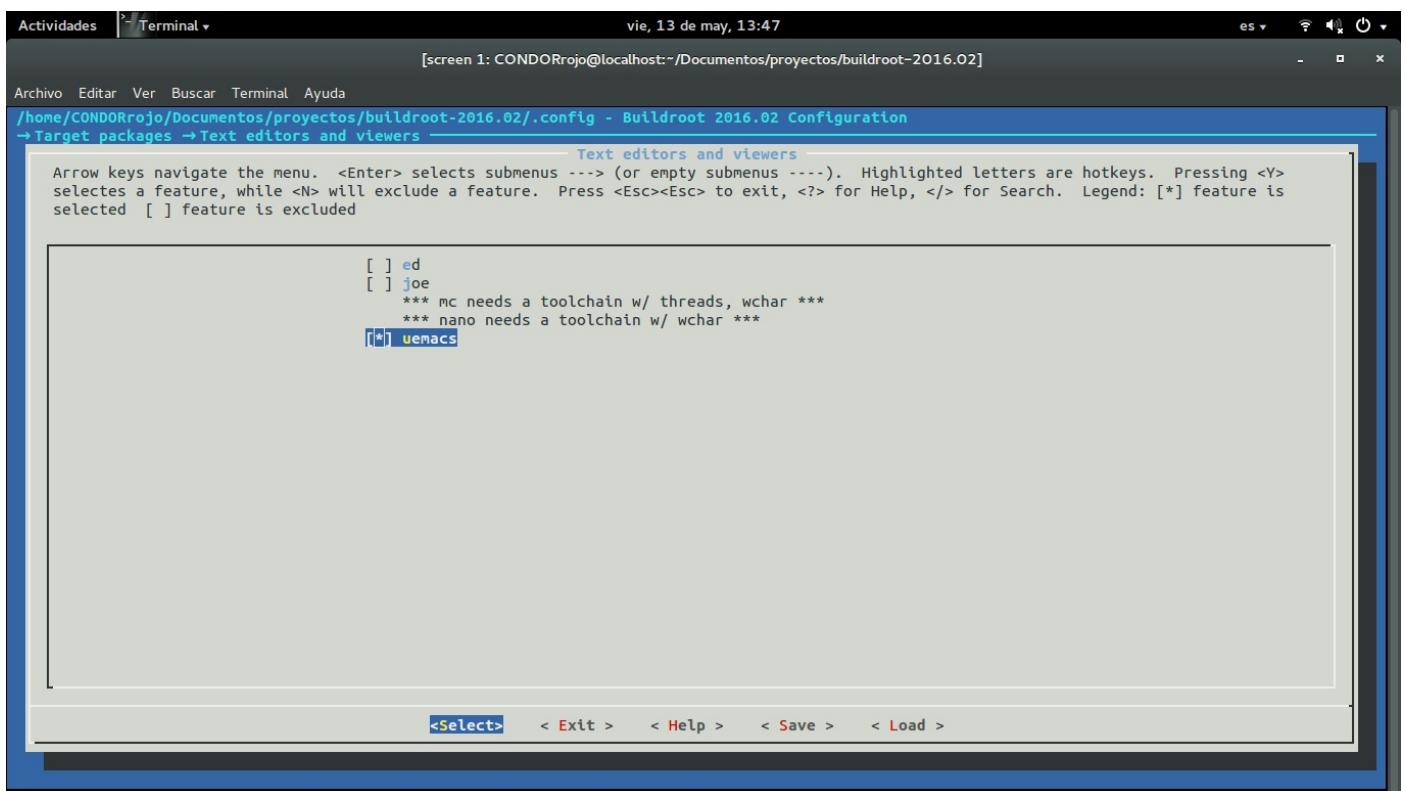
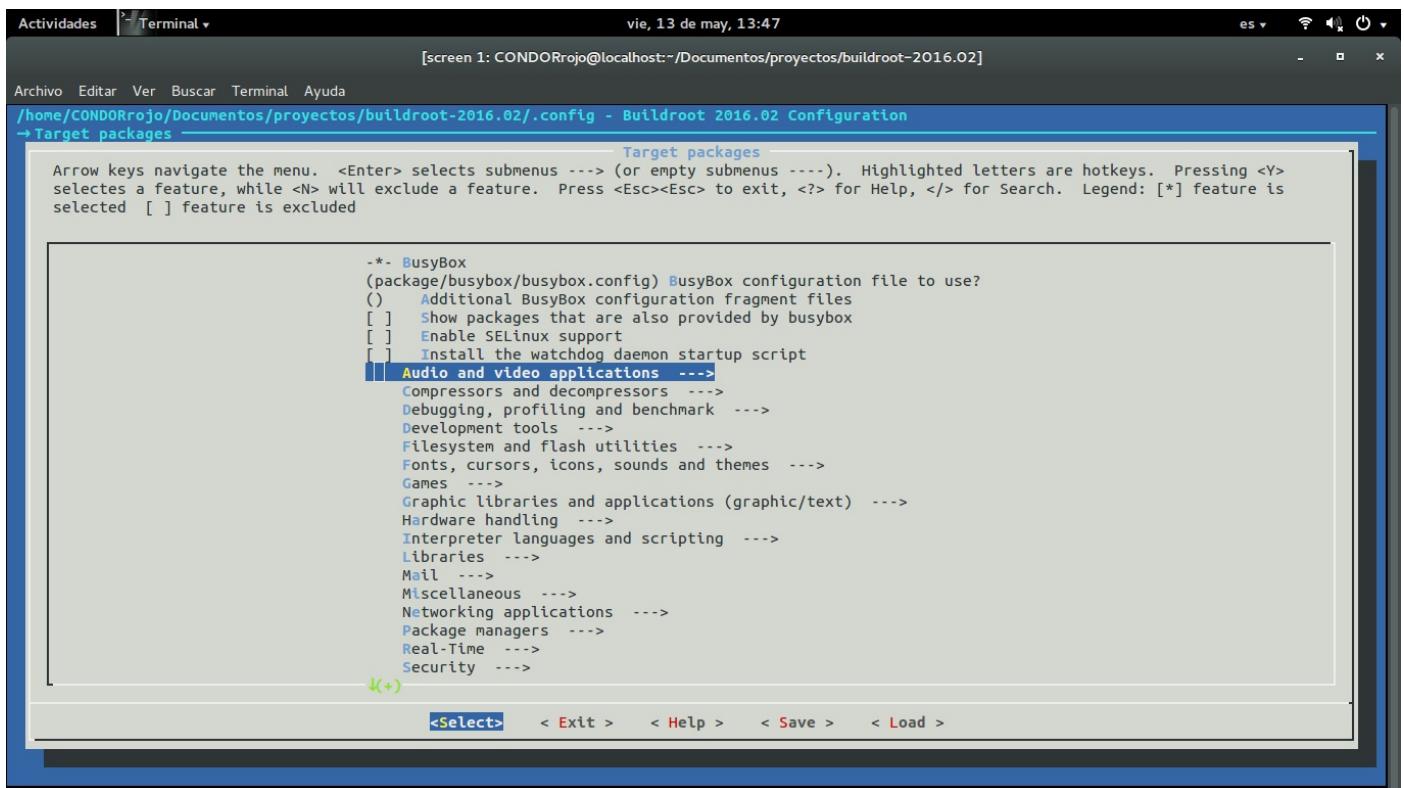


Para el "Kernel", podemos indicar de donde descargar el repositorio.



Otra opcion interesante es el "Target packages" donde podemos elegir paquetes que nos puedan ayudar en e objetivo de nuestro sistema

embebido. Como estoy habituado a usar emacs decidí instalar uemacs.



También podemos agregar lenguajes interpretador o scripting, en mi caso decidí instalar "micropython"

```
vie, 13 de may, 13:49  
[screen 1: CONDORrojo@localhost:~/Documentos/proyectos/buildroot-2016.02]  
Archivo Editar Ver Buscar Terminal Ayuda  
/home/CONDORrojo/Documentos/proyectos/buildroot-2016.02/.config - Buildroot 2016.02 Configuration  
→ Target packages → Interpreter languages and scripting  
Interpreter languages and scripting  
Arrow keys navigate the menu. <Enter> selects submenus --- (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> selects a feature, while <N> will exclude a feature. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] feature is selected [ ] feature is excluded  
[ ] enscript  
[ ] erlang  
[ ] gauche  
*** guile needs a uClibc or (e)glibc toolchain w/ threads, wchar, dynamic library ***  
[ ] haserl  
[ ] janvm  
[ ] jimgc  
[ ] lua  
[ ] luajit  
[*] micropython  
[ ] micropython-lib (NEW)  
[ ] moarvm  
[ ] mono  
*** nodejs needs a toolchain w/ C++, dynamic library, threads, gcc >= 4.8, wchar ***  
[ ] perl  
[ ] php  
*** python needs a toolchain w/ wchar, threads, dynamic library ***  
*** python3 needs a toolchain w/ wchar, threads, dynamic library ***  
*** ruby needs a toolchain w/ wchar, threads, dynamic library ***  
[ ] tcl  
  
<Select> < Exit > < Help > < Save > < Load >
```

Finalmente ejecutamos el comando make y esperamos un tiempo (en mi caso 40 min)

```
$ make
```

Pasado este tiempo obtenemos en la carpeta "output" tenemos los archivos que nos serviran para nuestro sistema embebido.

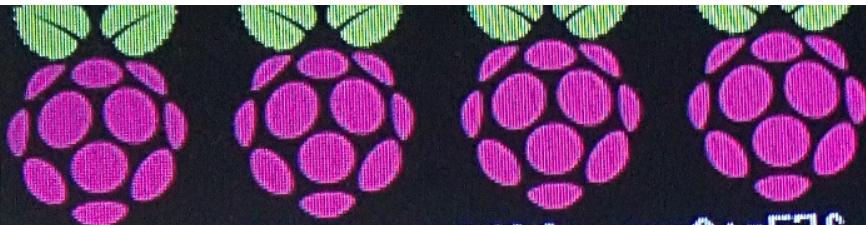
Basta con copiar el archivo "sdcard.img" a una memoria SD sin formato. Esto lo podemos hacerlo mediante el comando "dd" ejecutandolo desde el directorio que contiene la imagen de la siguiente manera:

```
sudo dd if=sdcard.img of=/dev/sdb
```

## Arranque del sistema

Ahora con la SD lista ya podemos arrancar nuestro sistema embebido

en la pantalla tenemos esto:



```
[ 1.145495] mmcblk0: mmc0:c57f SU04G 3.69 GiB
[ 1.145781] Indeed it is in host mode hprt0 = 000
[ 1.154366] mmcblk0: p1 p2
[ 1.227160] EXT4-fs (mmcblk0p2): couldn't mount as
[ 1.231987] EXT4-fs (mmcblk0p2): couldn't mount as
[ 1.243349] EXT4-fs (mmcblk0p2): mounted filesystem
[ 1.246889] VFS: Mounted root (ext4 filesystem)
[ 1.254624] devtmpfs: mounted
[ 1.259017] Freeing unused kernel memory: 420K (0)
[ 1.325673] usb 1-1: new high-speed USB device n
[ 1.329483] Indeed it is in host mode hprt0 = 000
[ 1.372552] EXT4-fs (mmcblk0p2): re-mounted. Opt
Starting logging: OK
Initializing random number generator... [ 1.4632
done.
Starting network...
[ 1.526125] usb 1-1: New USB device found, idVendor=0
[ 1.529970] usb 1-1: New USB device strings: Mfr=0
[ 1.534955] hub 1-1:1.0: USB hub found
[ 1.540116] hub 1-1:1.0: 5 ports detected
Bienvenido a mi SistemaEmebido
buildroot login: [ 1.815651] usb 1-1.1: new high-speed USB device
[ 1.915998] usb 1-1.1: New USB device found, idVendor=0
[ 1.919839] usb 1-1.1: New USB device strings: Mfr=0
[ 1.926574] smsc95xx v1.0.4
[ 1.989850] smsc95xx 1-1.1:1.0 eth0: register
[ 2.085646] usb 1-1.4: new low-speed USB device
[ 2.227017] usb 1-1.4: New USB device found, idVendor=0
[ 2.231028] usb 1-1.4: New USB device strings: Mfr=0
[ 2.235020] usb 1-1.4: Product: Dell USB Mouse
[ 2.239270] usb 1-1.4: Manufacturer: Dell
[ 2.257861] input: Dell Dell USB Mouse as /dev
[ 2.262590] hid-generic 0003:413C:3200.0001: i
[ 2.262590] hid-generic 0003:413C:3200.0001: i
```

```
[ 2.345651] usb 1-1.5: new low-speed USB device found,
[ 2.480923] usb 1-1.5: New USB device found,
[ 2.485287] usb 1-1.5: New USB device strings
[ 2.489871] usb 1-1.5: Product: DELL USB Keyb
[ 2.494215] usb 1-1.5: Manufacturer: DELL
[ 2.510770] input: DELL DELL USB Keyboard as
[ 2.565929] hid-generic 0003:413C:2005.0002:
[ 18.002666] random: nonblocking pool is initi
```

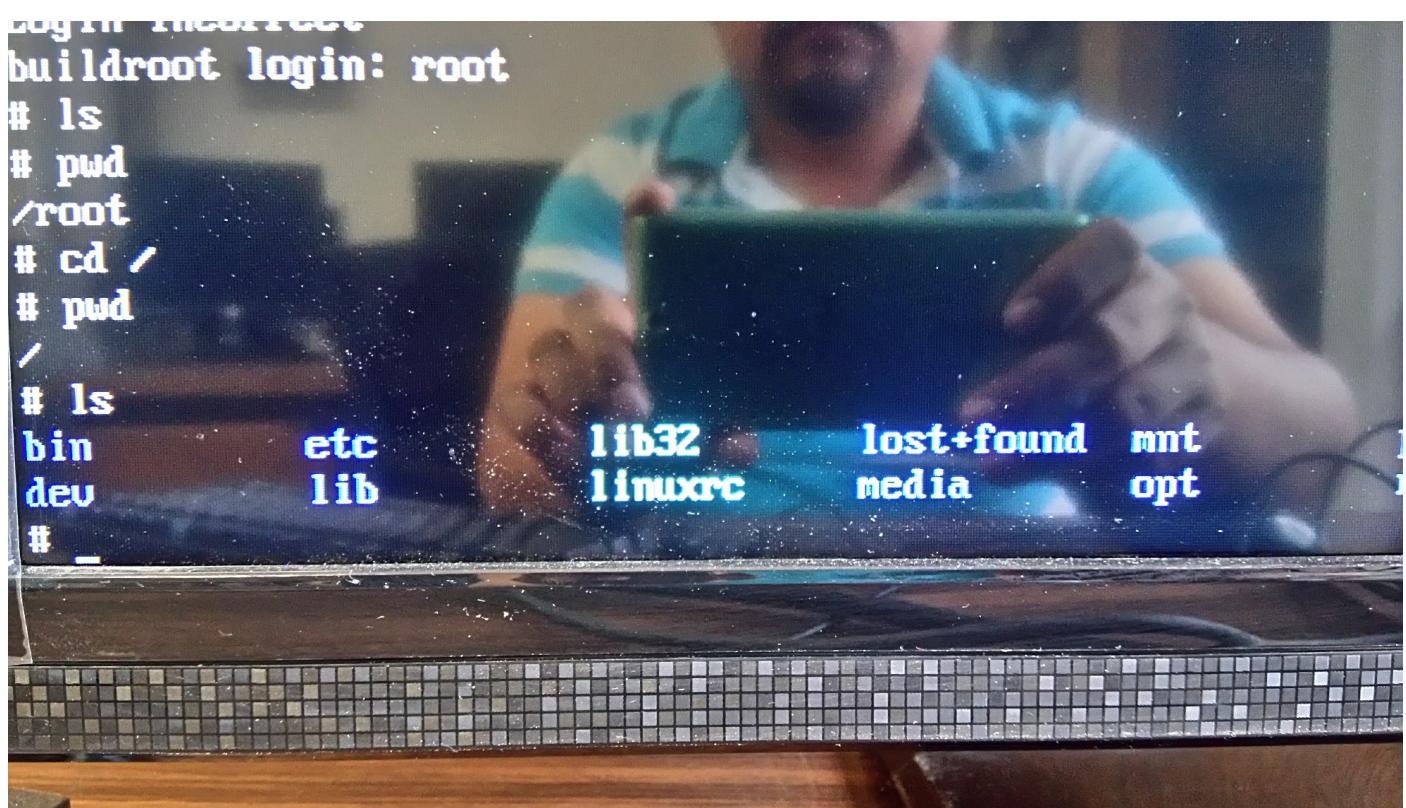
Se puede apreciar el banner personalizado:

```
1.354255] hub 1-1:1.0: USB device strings:
1.540116] hub 1-1:1.0: USB hub found
1.540116] hub 1-1:1.0: 5 ports detected
venido a mi SistemaEmbebido
root login: [ 1.815651] usb 1-1.1: new
1.915998] usb 1-1.1: New USB device found,
1.919839] usb 1-1.1: New USB device string
1.926574] sns950x v1.0.4
1.989850] sns950x 1-1.1:1.0 eth0: registered
2.085646] usb 1-1.4: new low-speed USB dev
2.227017] usb 1-1.4: New USB device found,
2.231028] usb 1-1.4: New USB device string
2.235020] usb 1-1.4: Product: Dell USB Mou
2.239270] usb 1-1.4: Manufacturer: Dell
2.257861] input: Dell Dell USB Mouse as /d
2.262590] hid-generic 0003:413C:3200.0001:
2.345651] usb 1-1.5: new low-speed USB dev
2.480923] usb 1-1.5: New USB device found,
2.485287] usb 1-1.5: New USB device strings
2.489871] usb 1-1.5: Product: DELL USB Keyb
2.494215] usb 1-1.5: Manufacturer: DELL
2.510770] input: DELL DELL USB Keyboard as
```

```
2.565929] hid-generic 0003:413C:2005.0002:  
8.002666] random: nonblocking pool is initi  
C[B[A[D  
ord:  
  
incorrect  
root login: pi  
ord:  
incorrect  
root login: root
```

En el siguiente archivo se presenta el arbol de directorios

```
login: incorrect  
buildroot login: root  
# ls  
# pwd  
/root  
# cd /  
# pwd  
/  
# ls  
bin etc lib32 lost+found mnt  
dev lib linuxrc media opt  
# _
```

A photograph of a man with dark hair and a beard, wearing a light blue polo shirt, holding a black tablet computer. He is looking at the screen of the tablet. The tablet displays a terminal window with the same text as the one above. The background shows a blurred indoor environment.

Tambien podemos ver informacion del espacio usado

```
# cd /  
# pwd  
/  
# ls  
bin      etc      lib32    lost+found  mnt      proc  
dev      lib      linuxrc   media       opt      root  
# df  
Filesystem           1K-blocks      Used   Available Use% Mounted on  
/dev/root            57869     49938      3783    93% /  
deutmpfs             451436        0     451436    0% /dev  
tmpfs                455744        0     455744    0% /dev/shm  
tmpfs                455744       28     455716    0% /tmp  
tmpfs                455744       16     455728    0% /run  
#
```

Y tambien informacion del sistema

```
# uname -a  
Linux buildroot 4.1.15-v7 #1 SMP PREEMPT Fri May 13 14:34:16 CDT 2016 armv7l  
# u  
udhcpc uevent umount uname uniq unix2dos unlink unlzma  
# u  
udhcpc uevent umount uname uniq unix2dos unlink unlzma  
# micro  
microcom micropython  
# m  
makedevs mdev microcom mkdir mknod mktemp  
md5sum mesg micropython mkfifo mkswap modprobe  
# e  
echo egrep eject em enu ether-wake expr  
# n  
nameif netstat nice nohup nslookup  
# v  
vconfig vi vlock  
#
```

¡Ahora ya podemos probar nuestro sistema embebido!