

Sistemas Críticos

Tema 1:

Selección y configuración de un sistema operativo

Lección 4:

Arranque de la placa y ejecución de *Linux*



Contenidos

Tema 1: Selección y configuración de un sistema operativo

Introducción

Fundamentos de *Linux*

Selección de la plataforma y prerequisites del sistema

Diseño de una plataforma de ejecución mínima

Construcción del *kernel* de *Linux*

Construcción del *Device Tree Blob*

Necesidad de un *Root File System*

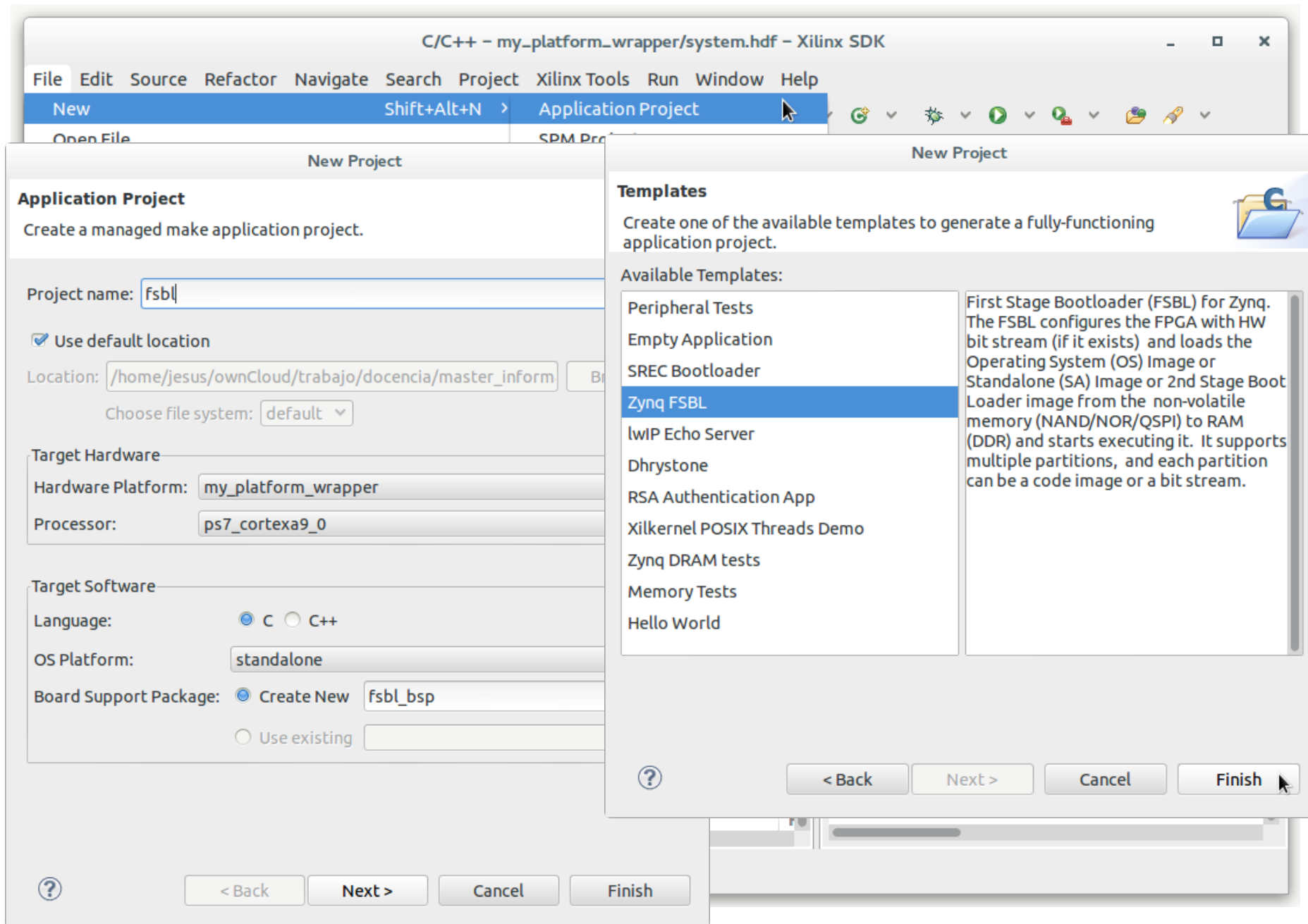
Construcción de un *Root File System*

Generación del *First Stage Boot Loader*

Construcción de *U-Boot*

Preparación de la imagen de arranque

Creación del *First Stage Boot Loader* mediante el SDK



Creación del *First Stage Boot Loader* mediante *scripts*

Variables de entorno

```
export PLATFORM="xilinx_zynq_a9"  
export PLATFORM_DIR="${PRJ_ROOT}/${PLATFORM}"  
export SDK_DIR="${PRJ_ROOT}/sdk"  
export PLATFORM_WRAPPER="${PLATFORM}_wrapper"  
export PLATFORM_WRAPPER_DIR="${SDK_DIR}/${PLATFORM_WRAPPER}"  
export FSBL="fsbl"  
export FSBL_DIR="${SDK_DIR}/${FSBL}"
```

Creación del FSBL

```
mkdir -p ${PLATFORM_WRAPPER_DIR}  
cp ${PLATFORM_DIR}/${PLATFORM}.runs/impl_1/${PLATFORM_WRAPPER}.sysdef \  
  ${PLATFORM_WRAPPER_DIR}/${PLATFORM_WRAPPER}.hdf  
hsi -mode batch -source fsbl.tcl  
cp ${FSBL_DIR}/executable.elf ${PRJ_ROOT}/images/fsbl.elf
```

Fichero `fsbl.tcl`

```
open_hw_design $env(PLATFORM_WRAPPER_DIR)/$env(PLATFORM_WRAPPER).hdf  
generate_app -hw $env(PLATFORM)_imp -os standalone -proc ps7_cortexa9_0 \  
  -app zynq_fsbl -compile -sw $env(FSBL) -dir $env(FSBL_DIR)
```

Contenidos

Tema 1: Selección y configuración de un sistema operativo

Introducción

Fundamentos de *Linux*

Selección de la plataforma y prerequisites del sistema

Diseño de una plataforma de ejecución mínima

Construcción del *kernel* de *Linux*

Construcción del *Device Tree Blob*

Necesidad de un *Root File System*

Construcción de un *Root File System*

Generación del *First Stage Boot Loader*

Construcción de *U-Boot*

Preparación de la imagen de arranque

Descarga y construcción de *U-Boot*

Variables de entorno

```
UBOOT="u-boot-Digilent-Dev"  
DILIGENT_GIT="https://github.com/DigilentInc"  
UBOOT_DIR="${PRJ_ROOT}/${UBOOT}"
```

Obtención de las fuentes

```
git -C ${PRJ_ROOT} clone -b master-next ${DILIGENT_GIT}/${UBOOT}.git
```

Limpiamos restos de compilaciones anteriores

```
cd ${UBOOT_DIR}  
make distclean
```

Configuramos y construimos

```
make zynq_zybo_config  
make -j 4
```

Copiamos el ejecutable de *U-Boot* al directorio de las imágenes

```
cp ${UBOOT_DIR}/u-boot ${PRJ_ROOT}/images/u-boot.elf
```

Contenidos

Tema 1: Selección y configuración de un sistema operativo

Introducción

Fundamentos de *Linux*

Selección de la plataforma y prerequisites del sistema

Diseño de una plataforma de ejecución mínima

Construcción del *kernel* de *Linux*

Construcción del *Device Tree Blob*

Necesidad de un *Root File System*

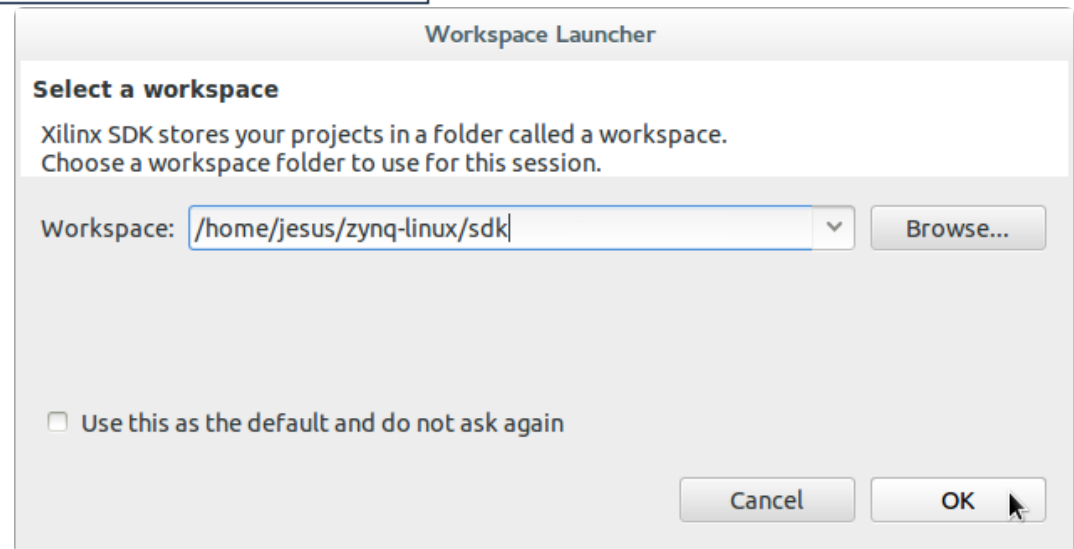
Construcción de un *Root File System*

Generación del *First Stage Boot Loader*

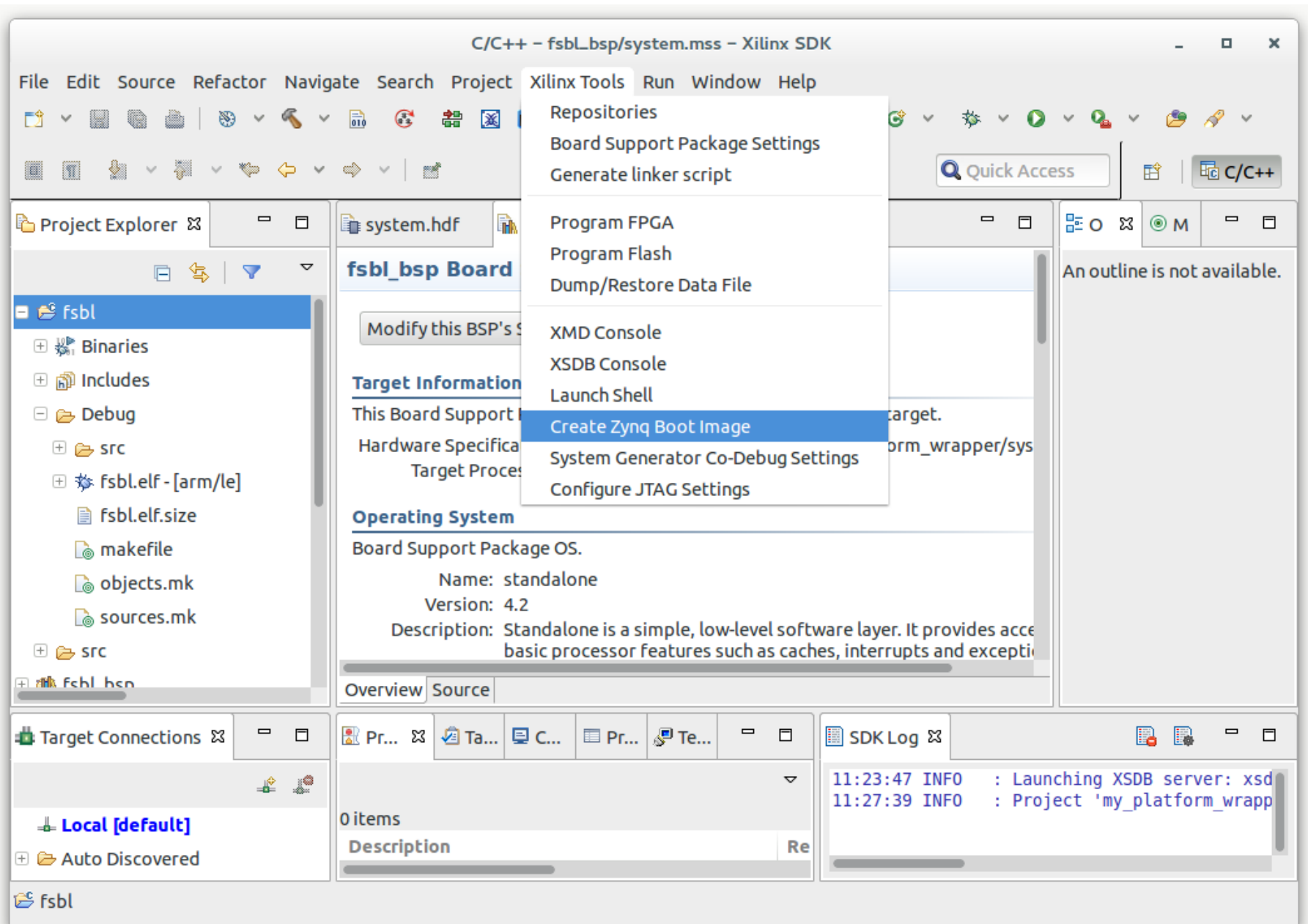
Construcción de *U-Boot*

Preparación de la imagen de arranque

Generamos la imagen de arranque de la placa mediante el SDK



Creamos la imagen de arranque



Añadimos U-Boot a la imagen

Create Zynq Boot Image

Creates Zynq Boot Image in .bin and .mcs formats from given FSBL elf and partition files in specified output folder.

☒ Create new BIF file ☐ Import from existing BIF file

Output BIF file path:

☐ Use Authentication

Authentication keys

PPK:

SPK:

SPK signature:

☐ Use encryption

Encryption key:

Key file:

Key store: ☒ BRAM ☐ EFUSE

Part name:

Boot image partitions

File path
(bootloader) /home/jesus/zynq
/home/jesus/zynq-linux/sdk/my

Output path:

Add partition

Add new boot image partition

File path:

Partition type:

Authentication: Encryption:

Checksum:

Presign:

Other

Alignment: Offset:

Reserve: Load:

Startup:

nticated	<input type="button" value="Add"/>
	<input type="button" value="Delete"/>
	<input type="button" value="Edit"/>
	<input type="button" value="Up"/>
	<input type="button" value="Down"/>

Generamos la imagen de arranque

Create Zynq Boot Image

Create Zynq Boot Image

Creates Zynq Boot Image in .bin and .mcs formats from given FSBL elf and partition files in specified output folder.

☒ Create new BIF file ☐ Import from existing BIF file

Output BIF file path:

Browse

☐ Use Authentication

Authentication keys

PPK:

Browse

 PSK:

Browse

SPK:

Browse

 SSK:

Browse

SPK signature:

Browse

☐ Use encryption

Encryption key:

Key file:

Browse

Key store: ☒ BRAM ☐ EFUSE

Part name:

Boot image partitions

File path	Encrypted	Authenticated
(bootloader) /home/jesus/zynq-linux/sdk/fsbl/Debug/fsbl.elf	none	none
/home/jesus/zynq-linux/sdk/my_platform_wrapper/xilinx_zynq_a9_wrapper.bit	none	none
/home/jesus/zynq-linux/u-boot-xlnx/u-boot.elf	none	none

Add

Delete

Edit

Up

Down

Output path:

Browse

?

Preview BIF Changes

Cancel

Create Image

Generamos la imagen de arranque de la placa mediante *scripts*

Variables de entorno

```
export PLATFORM="xilinx_zynq_a9"  
export SDK_DIR="${PRJ_ROOT}/sdk"  
export PLATFORM_WRAPPER="${PLATFORM}_wrapper"  
export PLATFORM_WRAPPER_DIR="${SDK_DIR}/${PLATFORM_WRAPPER}"
```

Copiamos el *bitfile* de la plataforma al directorio de las imágenes

```
cp ${PLATFORM_WRAPPER_DIR}/${PLATFORM_WRAPPER}.bit ${PRJ_ROOT}/images
```

Generamos la imagen de arranque

```
cd ${PRJ_ROOT}/images  
bootgen -image boot.bif -o boot.bin
```

Fichero `boot.bif` (define el formato de la imagen de arranque de la placa)

```
image :  
{  
    [bootloader]fsbl.elf  
    xilinx_zynq_a9.bit  
    u-boot.elf  
}
```

Preparación del *kernel* y el *RootFS* para ser cargados por *U-Boot*

Variables de entorno

```
UBOOT="u-boot-Digilent-Dev"  
UBOOT_DIR="${PRJ_ROOT}/${UBOOT}"  
KERNEL="Linux-Digilent-Dev"  
KERNEL_DIR="${PRJ_ROOT}/${KERNEL}"
```

Generamos una *ulmage* del *kernel* de *Linux*

```
PATH=$PATH:${UBOOT_DIR}/tools  
cd ${KERNEL_DIR}  
make -j 4 UIMAGE_LOADADDR=0x8000 uImage  
cp arch/arm/boot/uImage ${PRJ_ROOT}/images
```

Preparamos la imagen del *RootFS* para que pueda ser cargada por *U-Boot*

```
cd ${PRJ_ROOT}/images  
mkimage -A arm -T ramdisk -C gzip -d rootfs.cpio.gz uramdisk.image.gz
```

Preparamos la placa

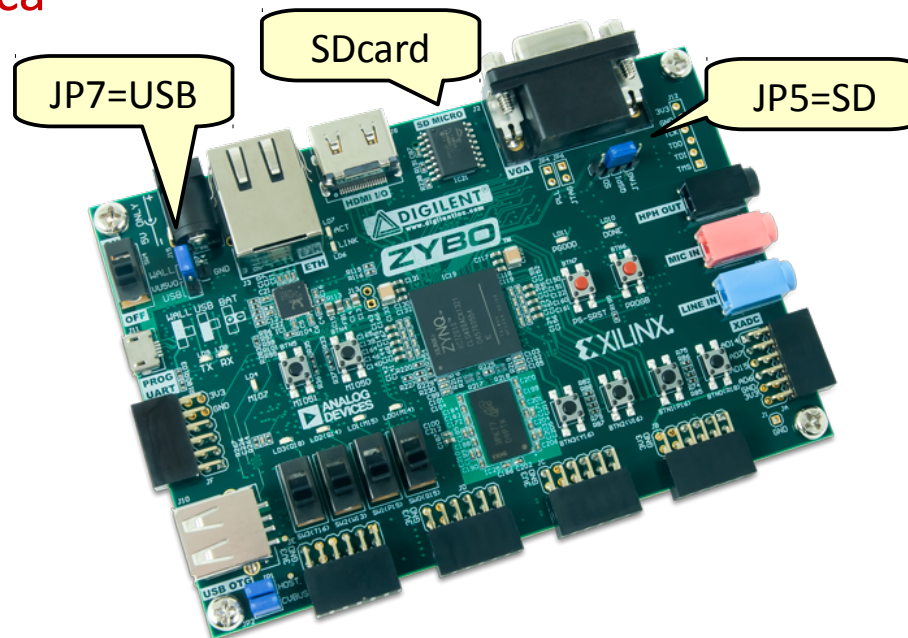
Variables de entorno

```
SDCARD_DIR="/media/jesus/9016-4EF8"
```

Configuramos la tarjeta microSD (copiamos a la primera partición)

```
cd ${PRJ_ROOT}/images  
cp boot.bin ${SDCARD_DIR}  
cp ${PRJ_ROOT}/images/uImage ${SDCARD_DIR}  
cp ${PRJ_ROOT}/images/uramdisk.image.gz ${SDCARD_DIR}  
cp ${PRJ_ROOT}/images/devicetree.dtb ${SDCARD_DIR}
```

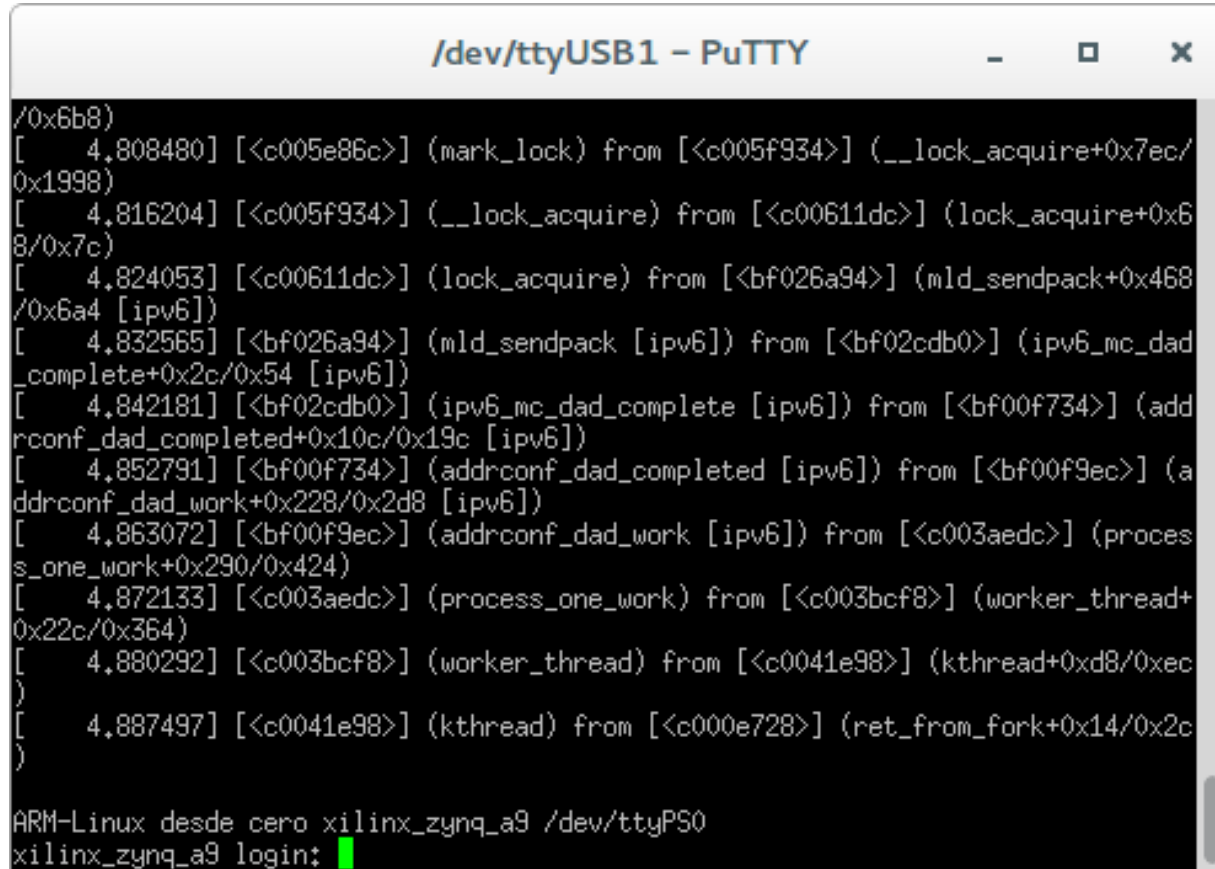
Preparamos la placa



Encendemos la placa y nos conectamos vía serie

Conexión a la placa

```
putty -serial -sercfg 115200 /dev/ttyUSB1
```



```
/dev/ttyUSB1 - PuTTY
/0x6b8)
[ 4,808480] [<c005e86c>] (mark_lock) from [<c005f934>] (__lock_acquire+0x7ec/
0x1998)
[ 4,816204] [<c005f934>] (__lock_acquire) from [<c00611dc>] (lock_acquire+0x6
8/0x7c)
[ 4,824053] [<c00611dc>] (lock_acquire) from [<bf026a94>] (mld_sendpack+0x468
/0x6a4 [ipv6])
[ 4,832565] [<bf026a94>] (mld_sendpack [ipv6]) from [<bf02cdb0>] (ipv6_mc_dad
_complete+0x2c/0x54 [ipv6])
[ 4,842181] [<bf02cdb0>] (ipv6_mc_dad_complete [ipv6]) from [<bf00f734>] (add
rconf_dad_completed+0x10c/0x19c [ipv6])
[ 4,852791] [<bf00f734>] (addrconf_dad_completed [ipv6]) from [<bf00f9ec>] (a
ddrconf_dad_work+0x228/0x2d8 [ipv6])
[ 4,863072] [<bf00f9ec>] (addrconf_dad_work [ipv6]) from [<c003aedic>] (proces
s_one_work+0x290/0x424)
[ 4,872133] [<c003aedic>] (process_one_work) from [<c003bcf8>] (worker_thread+
0x22c/0x364)
[ 4,880292] [<c003bcf8>] (worker_thread) from [<c0041e98>] (kthread+0xd8/0xec
)
[ 4,887497] [<c0041e98>] (kthread) from [<c000e728>] (ret_from_fork+0x14/0x2c
)

ARM-Linux desde cero xilinx_zynq_a9 /dev/ttyPS0
xilinx_zynq_a9 login:
```

Lecturas recomendadas

Linux en plataformas de Xilinx:

Xilinx. *Getting Started. Overview of the Xilinx Zynq AP SoC Design Flow.*

<http://www.wiki.xilinx.com/Getting+Started>

Diligent. *Embedded Linux Hands-on Tutorial for the ZYBO*, julio 2014.

http://diligentinc.com/Data/Products/ZYBO/ZYBO-Embedded_Linux_Hands-on_Tutorial.pdf

Bankras.org. *Xilinx Vivado 2014.4 and the Diligent ZYBO tutorial.*

<http://www.bankras.net/radko/uncategorized/xilinx-vivado-2014-4-and-the-diligent-zybo-tutorial/>