Cypher: Software Embebido

v0.3

Febrero 2019

INTRODUCCIÓN

Este documento incluye los pasos necesarios para preparar el dispositivo para su funcionamiento una vez que ha sido ensamblado.

Instalación Software

Antes de iniciar la instalación de software se exportan directorios de uso regular con el terminal en el directorio de trabajo (en este caso 'LinuxBuild'):

```
#~/LinuxBuild/
export CC=`pwd`/gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
export DISK=/dev/sdb
export kernel_version=4.14.79-ti-r84.1
```

1. Descargar crosscompiler, u-boot, kernel y hashbank

La siguiente lista de comandos

```
##Download ARM gcc compiler
wget -c
https://releases.linaro.org/components/toolchain/binaries/6.4-2018.05/arm-linux-gnueabihf/gcc-lina
ro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz
tar xf gcc-linaro-6.4.1-2018.05-x86_64_arm-linux-gnueabihf.tar.xz

##Download uboot
#~/LinuxBuild/
git clone https://github.com/u-boot/u-boot
cd u-boot/
#~/LinuxBuild/u-boot
git checkout v2018.09-rc2 -b tmp
wget -c
https://rcn-ee.com/repos/git/u-boot-patches/v2018.09-rc2/0001-am335x_evm-uEnv.txt-bootz-n-fixes.pa
tch
wget -c
https://rcn-ee.com/repos/git/u-boot-patches/v2018.09-rc2/0002-U-Boot-BeagleBone-Cape-Manager.patch
patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
patch -p1 < 0002-U-Boot-BeagleBone-Cape-Manager.patch

## Download Kernel
#~/LinuxBuild/
git clone https://github.com/RobertCNelson/ti-linux-kernel-dev.git
cd ti-linux-kernel-dev/
#~/LinuxBuild/ti-linux-kernel-dev/
git checkout origin/ti-linux-4.14.y -b tmp

## Download Debian File system
#~/LinuxBuild/
wget -c https://rcn-ee.com/rootfs/eewiki/minfs/debian-9.5-minimal-armhf-2018-07-30.tar.xz
tar xf debian-9.5-minimal-armhf-2018-07-30.tar.xz

## Download Hashbank (requires Username and password)
#~/LinuxBuild/
git clone https://github.com/MegaHashCorp/Hashbank_Embedded
```

2. Programar u-boot y kernel, preparar microSD

u-boot

```
## Pre-compile u-boot
#~/LinuxBuild/
cd u-boot
#~/LinuxBuild/u-boot
make ARCH=arm CROSS_COMPILE=${CC} distclean
make ARCH=arm CROSS_COMPILE=${CC} am335x_evm_defconfig
## Make changes to u-boot files
#Change boot delay
sed -i 's/CONFIG_BOOTDELAY=2/CONFIG_BOOTDELAY=-2/g' .config
# Change max_hz to 48MHz
sed -i 's/20000000);/48000000);/g' drivers/spi/mxc_spi.c
#Change max number of bytes the spi can handle
sed -i 's/define MAX_SPI_BYTES 32/define MAX_SPI_BYTES 153600/g' cmd/spi.c
sed -i 's/(bus, cs, 1000000,/(bus, cs, 48000000,/g' cmd/spi.c
# Change pinmux to start spi and run PWM as gpio replacing mux.c file
cp ../Hashbank_Embedded/uboot/mux.c  board/ti/am335x/
# Give the u-boot commands replacing am335x_evm.h file
cp ../Hashbank_Embedded/uboot/am335x_evm.h include/configs/
## Compile u-boot and return to main directory
make ARCH=arm CROSS_COMPILE=${CC}
cd ..
```

Kernel

```
## Build kernel
#~/LinuxBuild/
cd ti-linux-kernel-dev
#~/LinuxBuild/ti-linux-kernel-dev/
./build_kernel.sh
## Make changes to kernel files
sed -i 's/spi-max-frequency = <24000000>;/spi-max-frequency = <48000000>;/g'
KERNEL/arch/arm/boot/dts/am335x-pocketbeagle.dts
sed -i 's/bufsiz = 4096/bufsiz = 500000/g' KERNEL/drivers/spi/spidev.c
sed -i 's/P1_36_default_pin: pinmux_P1_36_default_pin { pinctrl-single,pins = <
              AM33XX_IOPAD(0x0990, PIN_OUTPUT_PULLDOWN | INPUT_EN | MUX_MODE1)
/P1_36_default_pin: pinmux_P1_36_default_pin { pinctrl-single,pins = <
              AM33XX_IOPAD(0x0990,     PIN_OUTPUT_PULLUP     |     INPUT_EN     |     MUX_MODE7)/g'
KERNEL/arch/arm/boot/dts/am335x-pocketbeagle-common.dtsi
```

```
##Rebuild kernel
#~/LinuxBuild/ti-linux-kernel-dev/
./tools/rebuild.sh
cd ..
```

microSD

```
#~/LinuxBuild/
sudo dd if=/dev/zero of=${DISK} bs=1M count=10
sudo dd if=./u-boot/MLO of=${DISK} count=1 seek=1 bs=128k
sudo dd if=./u-boot/u-boot.img of=${DISK} count=2 seek=1 bs=384k
sudo sfdisk ${DISK} <<-__EOF__
4M,,L,*
__EOF__
sudo mkfs.ext4 -L rootfs ${DISK}1
sudo mkdir -p /media/rootfs/
sudo mount ${DISK}1 /media/rootfs/
sudo tar xfvp ./*-*-*-armhf-*/armhf-rootfs-*.tar -C /media/rootfs/
sync
sudo chown root:root /media/rootfs/
sudo chmod 755 /media/rootfs/
sudo sh -c "echo 'uname_r=${kernel_version}' >> /media/rootfs/boot/uEnv.txt"
sudo cp -v ./ti-linux-kernel-dev/deploy/${kernel_version}.zImage
/media/rootfs/boot/vmlinuz-${kernel_version}
sudo mkdir -p /media/rootfs/boot/dtbs/${kernel_version}/
sudo tar xfv ./ti-linux-kernel-dev/deploy/${kernel_version}-dtbs.tar.gz -C
/media/rootfs/boot/dtbs/${kernel_version}/
sudo tar xfv ./ti-linux-kernel-dev/deploy/${kernel_version}-modules.tar.gz -C /media/rootfs/
sudo sh -c "echo '/dev/mmcblk0p1  /  auto  errors=remount-ro  0  1' >> /media/rootfs/etc/fstab"
sync
```

3. Pasar los .deb files para el TPM y la antena
4. Pasar hashbank
5. Añadir overlay
6. Instalar .deb files
7. Mover archivos necesarios (nodejs, sqlite3, NFC)
8. Mover service files a systemd