BuildRoot

Author:

Armando Emmanuel Correa Amorelli

Zapopan, Jalisco, August 7, 2023

Table of Contents

Introduction

Buildroot is an open-source software tool that facilitates the process of building customized embedded Linux systems. It is primarily used for creating minimalistic and tailored Linux distributions for embedded devices such as single-board computers, routers, set-top boxes, and other embedded systems.

Buildroot automates the process of cross-compiling all the necessary components and packages required for a functional Linux system, including the kernel, bootloader, libraries, user-space applications, and configuration files. This allows developers to create a lightweight and optimized system tailored to their specific requirements, which is especially important for resource-constrained devices where memory and storage space are limited.

Objective

Configure and build a custom system with Buildroot. This system should use systemd, udev, bash and be able to connect through ssh.

Configure Buildroot

Using menuconfig configure Buildroot as follow

Target Options:

- ***Target Architecture:*** ARM (Little endian)
- ***Target Architecture Variant:*** cortex-A8

Toolchain:

- ***Toolchain type:*** external toolchain
- ***TToolchain:*** ARM 2021.07

System Configuration:

- ***System hostname:*** billbeaglebone
- ***Root password:*** enable
- ***Init system***: systemd

Kernel:

- ***Linux kernel:*** 6.0
- ***Defconfig:*** omap2plus
- ***Kernel binary format:*** zImage
- ***In-tree Device Tree Source File name:*** am335x-boneblack
- ***Needs Host OpenSSL:*** enable

Target Packages:

- ***BusyBox:*** enable
- ***Networking applications:***
  - ***Dropbear:*** enable

Bootloaders:

- ***U-boot:*** enable
- ***U-boot configuration:*** am335x_evm
- ***Build system:*** Kconfig
- ***U-boot version:*** 2022.04
- ***U-boot binary format:*** u-boot.img
- ***Install U-Boot SPL binary image:*** enable
- ***U-boot SPL/TPL binary image name:*** MLO

Creating overlay

USB network setup

Now that we have our system configured, we need to add an overlay to be able to use Ethernet over USB. For this, we create a new directory at the following path

`board/beagleboneblack/rootfs-overlay/etc/init.d/`

At this directory, we will create a new file named *`S30usbgadget`* that will contain the following

```
#!/bin/sh

# set -e

GADGET_DIR=/config/usb_gadget/g1
OLDPWD=$(pwd)

printf "Starting USB gadget: "

modprobe cppi41
modprobe musb-dsps
modprobe phy-am335x
modprobe libcomposite

mkdir /config
mount -t configfs none /config
mkdir ${GADGET_DIR}
cd ${GADGET_DIR}
echo "0x05e8" > idVendor
echo "0xa4a1" > idProduct
mkdir strings/0x409
```

```
echo "serialnumber" > strings/0x409/serialnumber
echo "manufacturer" > strings/0x409/manufacturer
echo "ECM Gadget" > strings/0x409/product
mkdir functions/ecm.usb0
mkdir configs/c.1
mkdir configs/c.1/strings/0x409
echo Conf 1 > configs/c.1/strings/0x409/configuration
echo 120 > configs/c.1/MaxPower
echo "f8:dc:7a:00:00:01" > functions/ecm.usb0/host_addr
ln –s functions/ecm.usb0 configs/c.1
echo musb-hdrc.0 > UDC
cd ${OLDPWD}

echo "OK"
```

IP configuration

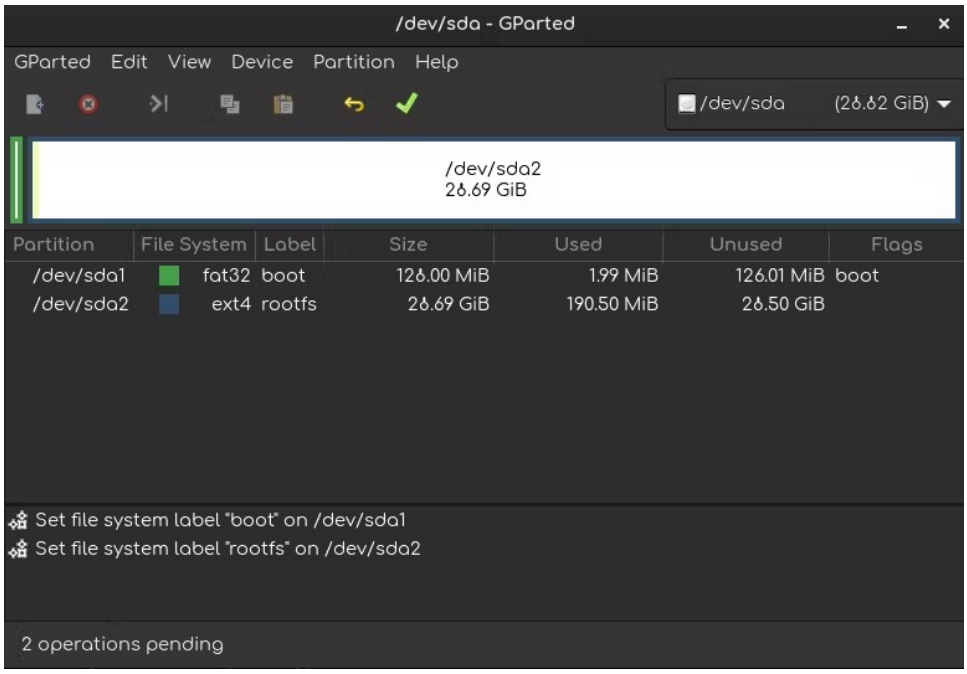
In `board/beagleboneblack/rootfs-overlay`, create a file named `etc/network/interfaces` with the following contents:

```
auto lo
iface lo inet loopback
auto usb0
iface usb0 inet static
address 192.168.0.2
netmask 255.255.255.0
```

Once we have all this, we can proceed with the build.

SD card preparation

For preparing the SD card, we will use Gparted, and create the following partitions and labels.



Flashing SD card

Now that we have build our system, and formatted the SD card, we need to flash it with the needed files.

First we copy the MLO zimage and other files to the boot partition

```
cp MLO u-boot.img zImage am335x-boneblack.dtb /media/$USER/boot/
```

Next we need to create a file that will specify to U-boot the kernel and device tree. This file should be at boot partition at the path `extlinux/extlinux.conf` and contain the following:

```
label buildroot
        kernel /zImage
        devicetree /am335x-boneblack.dtb
        append console=ttyO0,115200 root=/dev/mmcblk0p2 rootwait
```

Lastly, we need to decompress our file system at the rootfs partitions

```
sudo tar -C /run/media/$USER/rootfs/ -xf rootfs.tar
```

Setting up the BBB

Now that we have everything in our SD card, we proceed to boot the BBB from it. And make some last setups.

Starting the USB Network.

Thanks to the overlay that we made for the USB Network setup, we just need to run it

```
# /etc/init.d/S30usbgadget
Starting USB gadget: mkdir: can't create directory '/config': File exists
mount: /config: none already mounted on /config.
       dmesg(1) may have more information after failed mount system call.
mount: (hint) your fstab has been modified, but systemd still uses
       the old version; use 'systemctl daemon-reload' to reload.
mkdir: can't create directory '/config/usb_gadget/g1': File exists
mkdir: can't create directory 'strings/0x409': File exists
mkdir: can't create directory 'functions/ecm.usb0': File exists
mkdir: can't create directory 'configs/c.1': File exists
mkdir: can't create directory 'configs/c.1/strings/0x409': File exists
/etc/init.d/S30usbgadget: line 30: echo: write error: Device or resource busy
ln: configs/c.1/ecm.usb0: File exists
/etc/init.d/S30usbgadget: line 32: echo: write error: Device or resource busy
OK
```

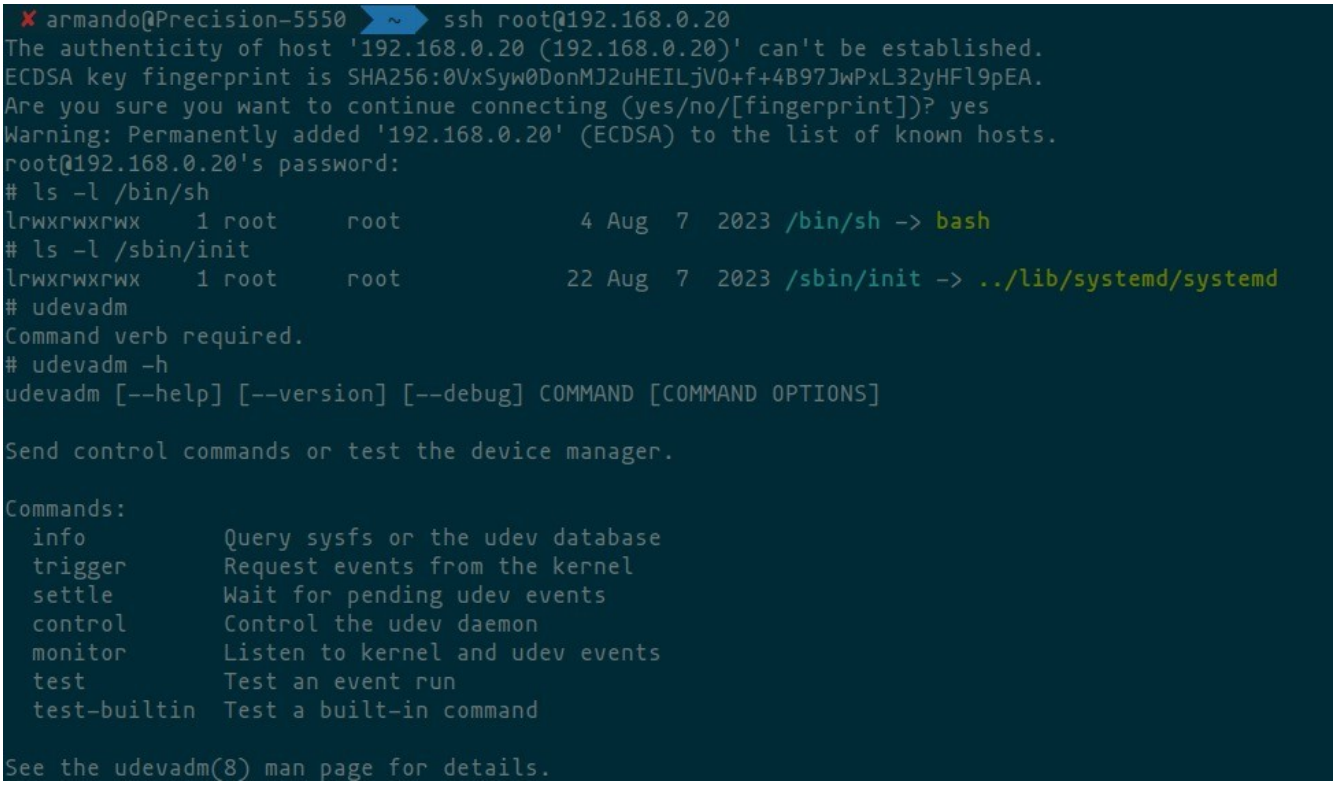
Getting the IP address ready

Same as with the USB Network we need to run a command to get the network interface up and get assigned a IP address

```
# ifup -a
ip: RTNETLINK answers: File exists
```

Connecting through ssh

Once we have done the last steps, we should be able to connect to the BBB from our Host by ssh and test that we actually are using bash, systemd and udev



Conclusions

This was a rather easier lab in comparison with all the ones made for the module 4, although I had some issues getting the connection between host and target working. For some reason I couldn’t ping the BBB from my laptop using the suggested IP (192.168.0.2), I needed to change it to 192.168.0.20, and it worked by art of magic.

Bibliography

[1] Buildroot Training - Bootlin, https://bootlin.com/doc/training/buildroot/buildroot-labs.pdf (accessed Aug. 8, 2023).

[2] B. V. Leeuwen, “Buildroot Embedded Linux beaglebone black,” Bill Van Leeuwen’s blog, https://blog.billvanleeuwen.ca/creating-a-minimal-linux-system-for-the-beaglebone-black-with-buildroot (accessed Aug. 7, 2023).