Report Overview

- Introduction to BeagleBone Black (BBB)
- Project Overview
- Host Setup
- Building & Deploying Linux on BBB
- Host/Target Development Environment
- Issues
- Miscellaneous Experiments
- Summary of Results
- References
Introduction

- About BeagleBone Black
- Additional Hardware
About BeagleBone Black

BeagleBone Black
1 GHz performance ready to use for $45

10/100 Ethernet
USB Host
Easily connects to almost any everyday device such as mouse or keyboard
microHDMI
Connect directly to monitors and TVs
microSD
Expansion slot for additional storage
512MB DDR3
Faster, lower power RAM for enhanced user-friendly experience

Expansion headers
Enable cape hardware and include:
- 66 digital I/O
- 7 analog
- 4 serial
- 2 SPI
- 2 I2C
- 8 PWMs
- 4 timers
- And much much more!

Power Button
Power Button
LEDS
Reset Button
USB Client
Development interface and directly powers board from PC

1 GHz Sitara AM335x ARM® Cortex™-A8 processor
Provides a more advanced user interface and up to 150% better performance than ARM11

2GB on-board storage using eMMC
- Pre-loaded with Ångström Linux Distribution
- 8-bit bus accelerates performance
- Frees the microSD slot to be used for additional storage for a less expensive solution than SD cards

Included in price:
- Power supply ~ $10
- USB network cable ~ $3
- 2GB on-board storage $5-$10
- PRU for real-time tasks typically on FPGA ~ $20
Additional Hardware Purchased

- D-link 5V 2.5A Power Supply
- Kingston 8GB microSDHC
- High Speed micro HDMI to HDMI cable
- USB to TTL Serial Cable (3.3V) FTDI Chip
- Edimax EW-7811Un Wireless USB Adapter
- BeagleBone Black Clear Compact Case
- Sabrent 4 port USB2.0 hub
Project Overview

- Goals
- Summary of results
Goals

- Download, build & deploy U-boot
- Download, build & deploy embedded Linux kernel for BBB
- Download, deploy root filesystem (Debian 7 Wheezy)
- Successfully deploy u-boot, kernel and rootfs using
  - Linked Host/Target Development setup
  - Removable storage Host/Target Development setup
- Configure/Build Linux kernel to support use of Edimax WLAN Adapter on BeagleBone Black
Summary of results

- **Built and deployed**
  - Crosstool-ng chain
  - U-boot bootloader
  - Linux Kernel (version 3.12.3) for BeagleBone Black

- **Deployed**
  - Debian 7 (Wheezy) root filesystem/distribution

- **Used the below Host/Target Development setups**
  - Linked (Ethernet + Serial Debug)
  - Removable storage (microSD)

- **Configured, rebuilt kernel for RTL8192CU device driver**
  - Enabled the Edimax EW7811 WLAN Adapter device
  - Device uses a Realtek chip which is compatible with RTL8192CU
Host Setup

- Environment
- Workspace setup
- Cross compile tool chain setup
Environment

- Operating System
  - Debian VM 3.2.46-1+deb7u1 i686 GNU/Linux
  - Oracle VM VirtualBox Manager Version 4.2.18 r88780

- Cross compiler Toolchain & GCC Version
  - crosstool-ng-1.19.0
  - gcc version 4.8.1 (output after crosstools are built)

- Additional Software/Packages installed
  - git, gtkterm, tftp, tftpd, xinetd, libncurses, etc..
  - FTDI USB/Serial Debug Cable Driver for Linux
  - Virtual Box Guest Additions
Workspace set up

- Create a project workspace
  - `cd ~`
  - `mkdir bbb/my_project_elinux`
  - `cd my_project_elinux`
  - `source create_project_layout` (creates dir. structure)
  - `source var_script` (sets environment variables)
Workspace Setup Shell Scripts

**create_project_layout**

```bash
#!/bin/bash
declare -a arr=(bootdir build-tools debug doc \ images kernel project rootfs sysapps)
for i in ${arr[@]}
do
  mkdir -p $i
  echo "Created $i"
done
```

**var_script**

```bash
#!/bin/bash
export PROJECT=my_project_elinux
export PRJROOT=/home/aditya/bbb/${PROJECT}
export TARGET=arm-unknown-linux
export HOST=i486-cross-linux-gnu
export PREFIX=${PRJROOT}/tools
export TARGET_PREFIX=${PREFIX}/${TARGET}
export PATH=${PREFIX}/bin:${PATH}
export CROSS_COMPILE=arm-none-linux-gnueabi-
cd $PRJROOT
```

```
aditya@debianvm:~/bbb/my_project_elinux$ ls
bootdir   create_project_layout  doc    kernel    rootfs    uEnv.txt
build-tools debug images  project  sysapps
aditya@debianvm:~/bbb/my_project_elinux$
```
Building the cross-compiler toolchain
(1 of 3)

- **Download** crosstool-NG and save the tar file in $PRJROOT/build-tools/
- Extract it and change into the new directory
  - run ./configure --prefix=$PRJROOT/tools/bin
  - make
  - make install
  - ct-ng help (Note: if it doesn't work, check your PATH)
  - ct-ng menuconfig
- Set the Target Architecture to arm
- Set the Target OS to linux
- Save the .config file & Exit menuconfig
- Build cross tools
  - ./ct-ng build
- After the toolchain is built, create a symbolic link
  - ln -s $PRJROOT/tools/gcc-/arm-/bin $PRJROOT/tools/bin
Building the cross-compiler toolchain

(2 of 3)

- Fully Built cross toolchain (GCC version 4.8.1)

```
aditya@debianvm:~/bbb/my_project_olinux/tools/bin$ ls -al
```
```bash
total 13376
-dr-xr-xr-x 2 aditya aditya  4696 Nov 18 07:36 .
dr-xr-xr-x 8 aditya aditya  4696 Nov 18 07:38 ..
-rwxr-xr-x 1 aditya aditya  616708 Nov 18 07:36 arm-unknown-linux-gnueabi-addr2line
-rwxr-xr-x 2 aditya aditya  643680 Nov 18 07:36 arm-unknown-linux-gnueabi-ar
-rwxr-xr-x 2 aditya aditya 1116872 Nov 18 07:36 arm-unknown-linux-gnueabi-as
-lrwxrwxrwx 1 aditya aditya   30 Nov 18 07:36 arm-unknown-linux-gnueabi-cc -> arm-unknown-linux-gnueabi-gcc
-rwxr-xr-x 1 aditya aditya  619204 Nov 18 07:36 arm-unknown-linux-gnueabi-c++filt
-rwxr-xr-x 1 aditya aditya  3603 Nov 17 23:01 arm-unknown-linux-gnueabi-ct-ng.config
-rwxr-xr-x 1 aditya aditya  22892 Nov 18 07:36 arm-unknown-linux-gnueabi-elfedit
-rwxr-xr-x 2 aditya aditya  677972 Nov 18 07:36 arm-unknown-linux-gnueabi-gcc
-rwxr-xr-x 2 aditya aditya  677972 Nov 18 07:36 arm-unknown-linux-gnueabi-gcc-4.8.1
-rwxr-xr-x 1 aditya aditya  22420 Nov 18 07:36 arm-unknown-linux-gnueabi-gcc-ar
-rwxr-xr-x 1 aditya aditya  22420 Nov 18 07:36 arm-unknown-linux-gnueabi-gcc-nm
-rwxr-xr-x 1 aditya aditya  22420 Nov 18 07:36 arm-unknown-linux-gnueabi-gcc-ranlib
-rwxr-xr-x 1 aditya aditya  305280 Nov 18 07:36 arm-unknown-linux-gnueabi-gcov
-rwxr-xr-x 1 aditya aditya  581540 Nov 18 07:36 arm-unknown-linux-gnueabi-gprof
-rwxr-xr-x 4 aditya aditya 1970632 Nov 18 07:36 arm-unknown-linux-gnueabi-ld
-rwxr-xr-x 4 aditya aditya 1970632 Nov 18 07:36 arm-unknown-linux-gnueabi-ld.bfd
-rwxr-xr-x 1 aditya aditya  10505 Nov 18 07:36 arm-unknown-linux-gnueabi-ldd
-rwxr-xr-x 2 aditya aditya  527940 Nov 18 07:36 arm-unknown-linux-gnueabi-nm
-rwxr-xr-x 2 aditya aditya  799316 Nov 18 07:36 arm-unknown-linux-gnueabi-objcopy
-rwxr-xr-x 2 aditya aditya  956836 Nov 18 07:36 arm-unknown-linux-gnueabi-objdump
-rwxr-xr-x 1 aditya aditya  10423 Nov 18 07:36 arm-unknown-linux-gnueabi-populate
-rwxr-xr-x 2 aditya aditya  643680 Nov 18 07:36 arm-unknown-linux-gnueabi-ranlib
-rwxr-xr-x 1 aditya aditya  315900 Nov 18 07:36 arm-unknown-linux-gnueabi-readelf
-rwxr-xr-x 1 aditya aditya  616628 Nov 18 07:36 arm-unknown-linux-gnueabi-size
-rwxr-xr-x 1 aditya aditya  617316 Nov 18 07:36 arm-unknown-linux-gnueabi-strings
-rwxr-xr-x 2 aditya aditya  789316 Nov 18 07:36 arm-unknown-linux-gnueabi-strip
aditya@debianvm:~/bbb/my_project_olinux/tools/bin$ ```
Building the cross-compiler toolchain (3 of 3)

- `./arm-none-linux-gnueabi-gcc -v`

```
gcc version 4.8.1 (crosstool-NG 1.19.0)
aditya@debianvm:~/bbb/my_project_elinux/tools/bin$  
```
Building & Deploying Linux on BBB

- Bootloader: U-boot
- Linux Kernel: v3.12
- Root file system
U-boot: download & apply patch

- **Download U-boot:**
  - cd ~/bbb/my_project_elinux/bootdir
  - git clone git://git.denx.de/u-boot.git
  - cd u-boot/
  - git checkout v2013.10 -b tmp

- **Apply patches:**
  - wget https://raw.githubusercontent.com/eewiki/u-boot-patches/master/v2013.10/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
  - patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
Configure and Build U-boot:

- cd ~/bbb/my_project_elinux/bootdir/u-boot
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabiv6 distclean
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabiv6-am335x_evm_config
- make ARCH=arm CROSS_COMPILE=arm-none-linux-gnueabiv6
U-boot: Download & patch log

aditya@debianvm:~/bbb/my_project_elinux/bootdir$ git clone git://git.denx.de/u-boot.git
Cloning into 'u-boot'...
remote: Counting objects: 240154, done.
remote: Compressing objects: 100% (47367/47307), done.
remote: Total 240154 (delta 192915), reused 235996 (delta 188982)
Receiving objects: 100% (240154/240154), 58.36 M1B | 455 K1B/s, done.
Resolving deltas: 100% (192915/192915), done.
Checking out files: 100% (0801/0801), done.
aditya@debianvm:~/bbb/my_project_elinux/bootdir$ cd u-boot
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ git checkout v2013.10 -b tmp
Checking out files: 100% (2063/2063), done.
Switched to a new branch 'tmp'
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ wget https://raw.githubusercontent.com/eewiki/u-boot-patches/master/v2013.10/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
--2013-12-06 08:41:47--  https://raw.githubusercontent.com/eewiki/u-boot-patches/master/v2013.10/0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 199.27.78.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|199.27.78.133|:443... HTTP request sent, awaiting response... 200 OK
Length: 5930 [5.8K] [text/plain]
Saving to: '0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch'
100%[=======================================] 5,930     --.-K/s in 0.001s

2013-12-06 08:41:52 (9.01 MB/s) - '0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch' saved [5930/5930]
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ patch -p1 < 0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
patching file arch/arm/cpu/armv7/am33xx/ddr.c
patching file include/configs/am335x_evm.h
patching file include/configs/ti_armv7_common.h
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$
U-boot: Configure & Build log

```bash
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ make ARCH=arm CROSS_COMPILE=${CC} distclean
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ make ARCH=arm CROSS_COMPILE=${CC} am335x_evm_config
Configuring for am335x_evm - Board: am335x_evm, Options: SERIAL1,CONS_INDEX=1,NAND
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ make ARCH=arm CROSS_COMPILE=${CC}
```

build U-Boot

```
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ ls
0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch
0001-am335x_evm-uEnv.txt-bootz-n-fixes.patch.1
api
arch
board
boards.cfg
common
config.mk
CREDITS
disk
doc
drivers
dts
```

1st stage bootloader

```
examples
fs
include
lib
Licenses
Makefile
mk-config
MKARCH
MKARCH
post
pread
tst
uboot
```

2nd stage bootloader

```
 ML0
 ML0.byteswap
 nand_swap
 net
 post
 u-boot.bin
 u-boot.lhs
 u-boot.lds
 u-boot.map
 u-boot.srec
```

```bash
aditya@debianvm:~/bbb/my_project_elinux/bootdir/u-boot$ ls
```

```
```
Linux Kernel: Download

- Download Linux Kernel
  - cd ~/bbb/my_project_elinux/kernel
  - git clone git://github.com/RobertCNelson/linux-dev.git
  - cd linux-dev

- Checkout v3.12x branch (latest)
  - git checkout origin/am33x-v3.12 -b tmp
Patch, configure & build

- cd ~/bbb/my_project_elinux/kernel/linux-dev
- ./build_kernel.sh

- This shell script patches the kernel
- Launches Menuconfig utility to configure the kernel &
- Builds the kernel and produces the below under ./deploy
  - Monolithic kernel image
  - Device tree
  - Firmware
  - Kernel modules

1st time build took ~6 hours
Linux Kernel Build Output

- **Build Directory**
  - Linux kernel directory
  ```
  aditya@debianvm:~/bbb/my_project_elinux/kernel/linux-dev$ ls
  build_deb.sh  ignore  patch.sh  system.sh
  build_kernel.sh  KERNEL  README  system.sh.sample
  deploy  LICENSE  repo_maintenance  tools
  dl  patches  scripts  version.sh
  ```

- **Build Output**
  - Fully built zImage, device tree, firmware and kernel modules
  ```
  aditya@debianvm:~/bbb/my_project_elinux/kernel/linux-dev/deploy$ ls -al
  total 17924
  drwxr-xr-x 2 aditya aditya  4096 Dec 6 11:05 .
  drwxr-xr-x 11 aditya aditya 4096 Dec 6 11:24 ..
  -rw-r--r-- 1 aditya aditya  114132 Dec 6 11:02 3.12.3-bone9.config
  -rw-r--r-- 1 aditya aditya  56008 Dec 6 11:05 3.12.3-bone9-dtbs.tar.gz
  -rw-r--r-- 1 aditya aditya 1222134 Dec 6 11:05 3.12.3-bone9-firmware.tar.gz
  -rw-r--r-- 1 aditya aditya 13708042 Dec 6 11:05 3.12.3-bone9-modules.tar.gz
  -rw-r-xr-x 1 aditya aditya  3238720 Dec 6 11:02 3.12.3-bone9.zImage
  ```
Root file system (Debian 7 Wheezy)

- Download & untar the rootfile system
  - `cd ~/bbb/my_project_elinux/rootfs`
  - `tar xf debian-7.1-bare-armhf-2013-08-25.tar.xz`

- Output
  - This file when untarred contains the rootfs. This will be copied into the microSD’s rootfs partition
Host/Target Development Setup

- Linked (TFTP netboot + Serial Debug)
- Removable Storage (microSD card)
Linked Development Setup

- Overview
- TFTP server setup
- Image transfer via TFTP
- Linux boot log
BeagleBone Black comes with Angstrom Linux (3.8.13, June 18th build) flashed on to eMMC.

Once booted the serial console shows:

```
Angstrom

The Angstrom Distribution beaglebone tty00

Angstrom v2012.12 - Kernel 3.8.13

beaglebone login: root

Last login: Sat Jan 1 00:00:20 UTC 2000 on tty00
root@beaglebone:~# uname -a
Linux beaglebone 3.8.13 #1 SMP Tue Jun 18 02:11:09 EDT 2013 armv71 GNU/Linux
root@beaglebone:~#
```
My goals were

- Rebuild uImage for Linux 3.8.13
- Transfer uImage via TFTP to BeagleBone Black
- Boot the newly built kernel
- Check for updated linux kernel version & build date

Indirectly I would also test

- TFTP capability
- Serial Debug console
- Ethernet connectivity
Building Linux kernel 3.8.13 (1 of 3)

- Download kernel source code
  - git clone git://github.com/beagleboard/kernel.git
  - cd kernel
  - git checkout 3.8 (checkout head of 3.8 release)
  - ./patch.sh

- Configure
  - cp configs/beaglebone
    kernel/arch/arm/configs/beaglebone_defconfig
  - wget http://arago-project.org/git/projects/?p=am33x-cm3.git;a=blob_plain;f=bin/am335x-pm-firmware.bin;hb=HEAD -O kernel/firmware/am335x-pm-firmware.bin
Build Kernel

- cd kernel
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf beaglebone_defconfig
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf uImage dtbs
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf uImage-dtb.am335x-boneblack
- make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf modules
Building Linux kernel 3.8.13 (3 of 3)

Kernel Build output

```
aditya@debianvm:~:bbb/my_project_elinux/kernel_old/kbuild/kernel/kernel/arch/arm/boot$ ls -al
total 19896
drwxr-xr-x 5 aditya aditya 4096 Nov 17 22:39 .
drwxr-xr-x 87 aditya aditya 4096 Nov 8 14:08 ..
drwxr-xr-x 2 aditya aditya 4096 Nov 8 13:22 bootp
drwxr-xr-x 2 aditya aditya 4096 Nov 8 23:21 compressed
drwxr-xr-x 3 aditya aditya 20480 Nov 8 23:49 dts
-rw-r--r-- 1 aditya aditya 46 Nov 8 13:22 .gitignore
-rw-r--r-- 1 aditya aditya 7109728 Nov 8 23:21 Image
-rw-r--r-- 1 aditya aditya 107 Nov 8 23:21 .Image.cmd
-rw-r--r-- 1 aditya aditya 1274 Nov 8 13:36 install.sh
-rw-r--r-- 1 aditya aditya 3404 Nov 8 13:53 Makefile
-rw-r--r-- 1 aditya aditya 4387424 Nov 8 23:21 uImage
-rw-r--r-- 1 aditya aditya 261 Nov 8 23:21 .uImage.cmd
-rw-r--r-- 1 aditya aditya 4412308 Nov 8 23:49 uImage-dtb.am335x-boneblack
-rw-r--r-- 1 aditya aditya 324 Nov 8 23:49 .uImage-dtb.am335x-boneblack.cmd
-rw-r--r-- 1 aditya aditya 4387360 Nov 8 23:21 zImage
-rw-r--r-- 1 aditya aditya 134 Nov 8 23:21 .zImage.cmd
aditya@debianvm:~:bbb/my_project_elinux/kernel_old/kbuild/kernel/kernel/arch/arm/boot$ ulmage (built on Nov 8 2013)
aditya@debianvm:~:bbb/my_project_elinux/kernel_old/kbuild/kernel/kernel/arch/arm/boot$
```
Setting up a TFTP server (1 of 4)

- Install tftp, tftpd, xinetd packages
  - sudo apt-get install xinetd tftp tftpd
- Create /etc/xinetd.d/tftp and put the below entry
  - sudo vi /etc/xinetd.d/tftp

```plaintext
service tftp
{
    disable = no
    protocol = udp
    port = 69
    socket_type = dgram
    wait = yes
    user = nobody
    server = /usr/sbin/in.tftpd
    server_args = /tftpboot
}
```
Setting up a TFTP server (2 of 4)

- Create a folder /tftpboot
  - `sudo mkdir /tftpboot`
  - `sudo chmod -R 777 /tftpboot`
  - `sudo chown -R nobody /tftpboot`

- Restart the xinetd service.
  - `sudo /etc/init.d/xinetd restart`

- Copy **ulimage** to /tftpboot
  - `sudo cp <a_long_path>/ulimage-dtb.am335x-boneblack /tftpboot/ulimage-bbb`
Setting up a TFTP server (3 of 4)

- `ls -al /tftpboot`

Folder within which files to be transferred are copied

The copied uImage-bbb file

```
root@debianvm:/# ls -al /tftpboot/
total 10656
drwxrwxrwx 2 nobody root 4096 Dec 7 09:03 .
- rw-r-xr-x 25 root root 4096 Dec 7 15:42 ..
-rwxrwxrwx 1 nobody root 21 Nov 20 11:43 test
-rw-r--r-- 1 root root 7 Dec 7 09:06 test.txt
-rwxrwxrwx 1 nobody root 4412308 Nov 9 17:15 uImage-bbb
-rw-r--r-- 1 aditya aditya 3238784 Dec 6 17:15 uImage-bbb-3.12
-rw-r--r-- 1 aditya aditya 3238784 Dec 6 18:22 uImage-new
root@debianvm:/# ```
Image Transfer via TFTP (1 of 3)

- Connect 5V DC power supply or mini USB cable
- Connect USB to Serial Debug cable
- Connect Ethernet Cable
- Launch gtkterm on Debian Host
  - Port = /dev/ttyUSB0
  - Speed = 115200
  - Parity = none
  - Flow control = none
  - Stopbits = 1
  - Bits = 8
  - Saved it as default configuration for quick restart
- Press Reset on BBB and you should start seeing boot messages in gtkterm
- **Note:** Install the Linux driver for the USB/Serial debug cable & connect via Devices -> USB menu in Virtual Box before launching gtkterm (The driver was available on FTDI chip’s website)
As soon as boot messages appear, press any key on your host to stop at U-Boot#
Image Transfer via TFTP (3 of 3)

- dhcp
- set client ipaddr
- Set server ip addr (ifconfig on host)
- Transfer uImage
- TFTP Transfer successfully completed
- TFTP Transfer complete
- bootm to start kernel
- Set bootargs
- Starting kernel
After Linux booted, I checked the updated kernel version and build date.

Updated Linux kernel 3.8.13, built on Nov 8 2013 (previous version was from June 2013).
Removable Storage setup

- Overview
- Set up microSD card
- Install u-boot, linux kernel & rootfs
- Boot Linux/Debian on BBB
- Linux boot log file
Overview

- My goals were to
  - Format & Partition microSD card
  - Copy files to boot and rootfs partitions
    - U-boot
    - Linux kernel, DTB, firmware
    - rootfs for Debian 7 (Wheezy)
  - Successfully boot Linux from microSD
Setup microSD card (1 of 3)

- Inserted microSD card in my laptop
- Connected to the microSD device using Devices -> USB Devices menu in Virtual Box
- Setup microSD card
  - `sudo`
  - `cd ~/bbb/my_project_elinux`
  - `lsblk`
  - `export DISK=/dev/sdb`
- **Note:** microSD card shows up as a `/dev/sdb` partition on my system
Setup microSD card (2 of 3)

- Erased microSD card
  - `dd if=/dev/zero of=${DISK} bs=1M count=64`

- Created partition layout
  - `sfdisk --in-order --Linux --unit M ${DISK} <<-__EOF__`
    - `1,48,0xE,*`
    - `*`
    - `__EOF__`
Setup microSD card (3 of 3)

- Formatted Partitions:
  - `sudo mkfs.vfat -F 16 ${DISK}1 -n boot`
  - `sudo mkfs.ext4 ${DISK}2 -L rootfs`

- Created Mount directories:
  - `sudo mkdir -p /media/boot/`
  - `sudo mkdir -p /media/rootfs/`

- For: DISK=/dev/sdb, mounted partitions
  - `sudo mount ${DISK}1 /media/boot/`
  - `sudo mount ${DISK}2 /media/rootfs/`
Install bootloader & bootscript

- Copied **MLO & u-boot.img** to the boot partition
  - cd ~/bbb/my_project_elinux/
  - sudo cp -v ./bootdir/u-boot/MLO /media/boot/
  - sudo cp -v ./bootdir/u-boot/u-boot.img /media/boot

- Copied uEnv.txt bootscript to the boot partition
  - cp uEnv.txt ~/bbb/my_project_elinux/bootdir/
  - sudo cp -v ./bootdir/u-boot/uEnv.txt /media/boot
Install rootfs

- Copied `rootfs` to the rootfs partition
  - `cd ~/bbb/my_project_elinux`
  - `sudo tar xfvp ./rootfs/*-*-armhf-*/*-armhf-rootfs-*/*.tar -C /media/rootfs/`
Install kernel files to ‘boot’ partition

- Copied Linux kernel files
  - cd ~/bbb/my_project_elinux/kernel
  - sudo cp -v ./linux-dev/deploy/3.12.3-bone9.zImage /media/boot/zImage

- Copied Linux kernel Device Tree Binaries
  - sudo mkdir -p /media/boot/dtbs/
  - sudo tar xfov ./linux-dev/deploy/3.12.3-bone9-dtbs.tar.gz -C /media/boot/dtbs/
  - sudo tar xfv ./linux-dev/deploy/3.12.3-bone9-firmware.tar.gz -C /media/rootfs/lib/firmware/
Install kernel modules to ‘rootfs’ partition

- Copy **Linux kernel modules**
  - `sudo tar xfv ./linux-dev/deploy/3.12.3-bone9-modules.tar.gz -C /media/rootfs/`

- Remove microSD card
  - `sync`
  - `sudo umount /media/boot`
  - `sudo umount /media/rootfs`
microSD card (boot partition)

- Boot partition & files copied to microSD card

![Image of boot partition and U-boot & Linux Kernel files]

- boot partition
- U-boot & Linux Kernel files

Files in the boot partition:
- dtbs
- MLO
- u-boot.img
- uEnv.txt
- zImage
microSD card (rootfs partition)

- rootfs partition & files copied to microSD card

- Debian 7 (Wheezy) rootfs files

- rootfs partition
Insert microSD card into BBB’s microSD card slot

Attach the following devices/cables to the BBB:
- microHDMI cable to microHDMI port & HDMI monitor
- mouse + keyboard via a USB hub to USB Host Port
- Ethernet cable to the Ethernet Port
- Serial Debug cable to the Debug Port & PC
- Mini USB cable to power the BBB

Within a minute or a little more your should see a login screen appear on the HDMI monitor where the user can login using:
- Login: debian
- Password: temppwd
Linux boot log on BBB (1 of 5)

Reading u-boot.img

microSD card found

reading zImage

Starting kernel
Linux boot log on BBB (2 of 5)

Linux version 3.12.3-bone9

Kernel command line: bootargs
USB 2.0 Hub found

Checking rootfs
Debian GNU/Linux 7 login prompt

cat /proc/*-release

uname –a (3.12.3-bone9 Linux ARM built on December 6 2013)
Wi-Fi over USB on BeagleBone Black

- Kernel Configuration for RTL8192CU
- Linux boot log
- Configuring Wi-Fi network connectivity on BBB
- Confirming Wi-Fi network connectivity on BBB
Kernel Configuration for RTL8192CU
(1 of 3)

- **Goal:** Configure Linux kernel to build the loadable kernel device driver module/firmware for RTL8192CU device
- cd ~/bbb/my_project_elinux/kernel/linux-dev
- ./build_kernel.sh
  - After a few mins, menuconfig should pop up
Kernel Configuration for RTL8192CU (2 of 3)

- Device Drivers -> Network device support -> Wireless LAN
Wireless LAN -> Realtek rtlwifi family of devices (Press ‘y’)

-> Realtek RTL8192CU/RTL8188CU USB Wireless Network Adapter (Press ‘y’)

- Exit & Save .config
- Wait until the kernel build completes
- Copy new kernel files to the microSD card (Refer Slide 46 & 47)
Setup BBB to boot Linux (Refer Slide 50)

Check Linux boot log messages for:
- WLAN over USB adapter instantiation
- Loading of rtl8192cu firmware

```
U-Boot SPL 2013.04-dirty (Jun 19 2013 - 09:57:14)
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Peripheral mode controller at 47401000 using PIO, IRQ 0
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
musb-hdrc: 28/31 max ep, 16384/16384 memory
USB Host mode controller at 47401800 using PIO, IRQ 0
MMC SD/MMC: 0
mmc_send_cmd: timeout: No status update
reading u-boot.img
reading u-boot.img

U-Boot 2013.04-dirty (Jun 19 2013 - 09:57:14)
I2C: ready
DRAM: 512 MiB
WARNING: Caches not enabled
NAND: No NAND device found!!!
  0 MiB
MMC: CMAF SD/MMC: 0, CMAF SD/MMC: 1
 *** Warning - readev() failed, using default environment
musb-hdrc: ConfigData=0xde (UTMI-8, dyn FIFOs, HB-ISO Rx, HB-ISO Tx, SoftConn)
musb-hdrc: MHDRC RTL version 2.0
musb-hdrc: setup fifo_mode 4
```
Importing environment from u-boot ...
Running uEnvcmd ...
reading zImage
3238720 bytes read in 373 ms (8.6 MByte/s)
reading /dtbs/am335x-boneblack.dtb
26024 bytes read in 12 ms (2.1 MByte/s)
## Flattened Device Tree blob at 80f80000
   Booting using the EDB blob at 0x80f80000
   Using Device Tree in place at 80f80000, end 80f895a7

Starting kernel ...

[0.000000] Booting Linux on physical CPU 0x0
[0.000000] Initializing cgroup subsys cpuset
[0.000000] Initializing cgroup subsys cpu
[0.000000] Initializing cgroup subsys cpuacct
[0.000000] Linux version 3.12.3-bone9 (edity@debianvm) (gcc version 4.8.2 20131014 (prerelease) (crosstool-NG linaro-1.19.1-4.8-2013.10 - Linaro GCC 2013.10) ) #1 SMP Fri Dec 6 10:33:17 EST 2013
[0.000000] CPU: ARMv7 Processor [113fc062] revision 2 (ARMv7), cr=1003367d
[0.000000] CPU: PVIPT / VPIPT nonaliasing data cache, VPIPT aliasing instruction cache
[0.000000] Machine: Generic AM33XX (Flattened Device Tree), model: TI AM335x BeagleBone
[0.000000] Memory policy: ECC disabled. Data cache writeback
[0.000000] CPU: All CPU(s) started in SVC mode.
[0.000000] AM335X ES2.0 (sgx neon )
[0.000000] PERCPU: Embedded 8 pages/cpu @0000000 s856 r8192 d14720 u32768
[0.000000] Built 1 zone lists in Zone order, mobility grouping on. Total pages: 128792
[0.000000] Kernel command line: console=tty0,115200n8 root=/dev/mmcblk0p2 ro rootfstype=ext4 rootwait fixrtc
[0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)
[0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[0.000000] allocated 1048576 bytes of page_cgroup
[0.000000] please try 'cgroup_disable=memory' option if you don't want memory cgroups
[0.000000] Memory: 507820K/523264K available (6048K kernel code, 454K rwdata, 2420K rodata, 257K init, 572K bss, 15444K reserved, OK highmem)
Realtek WLAN USB Adapter instanced

Loaded rtl8192cu firmware (compatible with Edimax)
Succesfully completed booting new Linux kernel with Edimax EW7811 WLAN USB Adapter device support
Configuring for Wi-Fi connectivity (1 of 3)

- lsusb
  - Executed in embedded Linux shell of BBB
  - Shows the Edimax EW-7811 WLAN Adapter device
- cd /lib/firmware/rtlwifi
  - Executed in embedded Linux shell of BBB
  - Shows the Realtek device firmware
Configuring for Wi-Fi connectivity (2 of 3)

- `sudo vi /etc/network/interfaces`
  - Executed in the embedded linux shell of BBB
  - Added the wlan0 interface commands as shown below

```
auto lo
defaul inet loopback
auto eth0
defaul inet dhcp

# WiFi setup
interface wlan0 inet dhcp
  wpa-ssid "<ENTER_YOUR_NW_NAME>"
  wpa-psk "<ENTER_YOUR_NW_PASSWD>
```

Edit interfaces file to enter the below commands, Save & Exit vi
Configuring for Wi-Fi connectivity (3 of 3)

- ifup wlan0
  - This command will bring up the wlan0 interface.

- wlan0 is authenticated and an IP address is assigned to BeagleBone Black
  - IP addr of BeagleBone Black – 192.168.1.140
Confirming Wi-Fi connectivity (1 and 2)

DHCPACK from 192.168.1.1
bound to 192.168.1.140 -- renewal in 35800 seconds.
root@arm:/lib/firmware/rtlwifi# ping google.com
PING google.com (74.125.226.196) 56(84) bytes of data.
64 bytes from lga15s28-in-f4.1e100.net (74.125.226.196): icmp_seq=1 ttl=55 time=20.1 ms
64 bytes from lga15s28-in-f4.1e100.net (74.125.226.196): icmp_seq=2 ttl=55 time=30.5 ms
64 bytes from lga15s28-in-f4.1e100.net (74.125.226.196): icmp_seq=3 ttl=55 time=199 ms

--- google.com ping statistics ---
6 packets transmitted, 4 received, 33% packet loss, time 9427 ms
rtt min/avg/max/mdev = 20.129/68.052/199.476/75.978 ms
root@arm:/lib/firmware/rtlwifi#
root@arm:/lib/firmware/rtlwifi#
root@arm:/lib/firmware/rtlwifi# at[Kpt-get install vim
Reading package lists... 0%
Reading package lists... 0%
Reading package lists... 1%
Reading package lists... 5%
Reading package lists... 9%
Reading package lists... 13%
Reading package lists... 17%
Reading package lists... 21%
Reading package lists... 24%

ping google.com
BBB is connected to my home network
package installation is progressing proving BBB WiFi connectivity
Confirming Wi-Fi connectivity (2 and 2)

vim was successfully installed on BBB
Summary of Issues
Issue # 1

Problem
- Linux Kernel takes too long to build
- ~6 hours for a clean build (i.e. not an incremental build)

Fix
- Modified the build script to use both cores at build time
  - vi ~/bbb/my_project_elinux/kernel/linux-dev/build_kernel.sh
  - Added ‘CORES=2’ in the make_kernel() subroutine
  - $CORES is a command line argument to make at build time

Impact
- Build time was cut down by 50%
- ~3 hours for a clean build (instead of 6 hours)
Problem
- TFTP connection between server and client was too slow
- The file transfer would time out
- Took me several hours to fix this issue

Fix
- Used DC power supply vs. USB supplied power
- Power cycled the router
- Used “Bridged” Network connection on Virtual Box

Impact
- Was able to successfully transfer ulmage using TFTP
- The transfer was more reliable and faster
Issue # 3

Problem

- How to convert a uImage from a zImage?
- I wanted to do a TFTP boot but I did not want to re-build the kernel to generate a non-compressed uImage

Fix

- Use mkimage utility to convert from zImage -> uImage
- E.g. mkimage -A arm -O linux -T kernel -C none -a 0x8020000 -e 0x80200000 d ./3.12.3-bone9.zImage ./tmp/uImage-3.12.3-bone9

Impact

- Saved ~3 hours of kernel re-build time
I tried these experiments during the course of this project

- Used pre-built crosstool chains
  - arm-linux-gnueabi-gcc4.4
  - arm-linux-gnueabi-gcc4.6
- Used Ubuntu VM Host environment (in place of Debian)
- Deployed Ubuntu distribution to BBB (in place of Debian)
- Built uImage by modifying the makefile (instead of zImage)
- Updated latest Angstrom Linux images on BBB
- Tried use of minicom without much luck/decided upon gtkterm
Summary

- Built and deployed
  - Crosstool-ng chain
  - U-boot bootloader
  - Linux Kernel (version 3.12.3) for BeagleBone Black

- Deployed
  - Debian 7 (Wheezy) root filesystem/distribution

- Used the below Host/Target Development setups
  - Linked (Ethernet + Serial Debug)
  - Removable storage (microSD)

- Configured, rebuilt kernel to support Edimax 7811 WLAN USB Adapter
List of references

- Building BBB Kernel - eLinux.org
- BeagleBone Black - Linux on ARM - eewiki
- BeagleBoneBlack Building Kernel – BeyondLogic
- Booting Ubuntu on BeagleBoard Black - Circuitco Wiki Support
- SSH on Mac and Linux | SSH to BeagleBone Black over USB | Adafruit Learning System
- Installing Emdebian ARM Cross Toolchain in Debian
- Getting Started with BeagleBone Black
- How do I install and run a TFTP server? - Ask Ubuntu
- http://derekmolloy.ie/beaglebone/
THANK YOU