Yocto for BBB.

### Build Host

BUILD HOST is based on the OpenEmbedded Project. This project uses bitbake to construct images.

Reference Build Host POKY = Bitbake + OpenEmbedded

The OpenEmbedded build system is able to run on Ubuntu that has the following versions for Git, tar, and Python.

Dependencies – OpenEmbedded

Git 1.7.8 or greater

tar 1.24 or greater

Python 2.7.3 or greater excluding Python 3.x, which is not supported.

Dependencies - Build Host Packages¶

These packages installed on the build host using an Ubuntu System

* bc
* libsdl1.2-dev
* build-essential
* pkg-config
* chrpath
* socat
* diffstat
* subversion
* gawk
* texi2html
* git
* texinfo
* libncurses5-dev
* u-boot-tools

Again, if you use Ubuntu, you can use sudo. Debian does not come with sudo, so just apt-get as you normally do - as superuser su whatever.

sudo apt-get install bc build-essential chrpath diffstat gawk git libncurses5-dev pkg-config socat subversion texi2html texinfo u-boot-tools

### Repositories

YOCTO RELEASE

Clone the Yocto Project repository poky jethro. Below is an example from an Ubuntu build host that clones the poky repository and then checks out the latest Yocto Project Release:

$ git clone -b jethro git://git.yoctoproject.org/poky.git poky-jethro

META-OPENEMBEDDED

Clone the OpenEmbedded repo.

$ git clone -b jethro git://git.openembedded.org/meta-openembedded

META-BEAGLEBOARD repository

Clone the official Beagleboard repo. Before cloning, create a sub directory and change to that directory.

mkdir bbb

cd bbb

$ git clone git://github.com/beagleboard/meta-beagleboard

### <http://www.jumpnowtek.com/beaglebone/BeagleBone-Systems-with-Yocto.html>

<https://www.hackster.io/30847/yocto-on-beaglebone-black-9ae649>

(Help links )

### Building Images¶

**$ mkdir -p ~/bbb/build/conf**

**vaddevam@vta069:~/BBB$ source poky-jethro/oe-init-build-env ~/bbb/build**

**vaddevam@vta069:~/bbb/build$ bitbake core-image-minimal**

echo 'MACHINE = "beaglebone"' >> conf/local.conf

echo 'PREFERRED\_VERSION\_linux-raspberrypi = "4.%” >> conf/local.conf

echo 'DISTRO\_FEATURES\_remove = "x11 wayland"' >> conf/local.conf

echo 'DISTRO\_FEATURES\_append = " systemd"' >> conf/local.conf

echo 'PARALLEL\_MAKE ?= " -j ${@oe.utils.cpu\_count()}"' >> conf/local.conf

echo 'BB\_NUMBER\_THREADS ?= " ${@oe.utils.cpu\_count()}"' >> conf/local.conf

$ sytemd-analyze > log.txt

$ sytemd-analyze time>

$ sytemd-analyze lamp

Now that you have your system requirements in order, you can give the Yocto Project a try. This section presents steps that let you do the following:

Initialize the Build Directory

The build directory can be either manually created or created using the oe-init-build-env script.

1. Manually - Configure these paths to the meta layers per your standards.

2. With the script oe-init-build-env

Customize the config files

When using oe-init-build-env to create the build directory, some configuration files are generated in the build/conf directory.

Editing bblayers.conf

The directory structure in bblayers.conf should look like this

~/poky-jethro/

meta-openembedded

~/bbb/

meta-beagleboard

build/

conf/

Editing local.conf

There are 3 variables to edit in local.conf

Editing these are not necessary but optional.

DL\_DIR – Where downloaded sources are stored

SSTATE\_DIR – This is usually 5gb and can be moved from home

TMP\_DIR – Where the temporary build and final binaries are found

### Initialize the Build Environment:

Run the oe-init-build-env environment setup script to use bitbake. This defines the OpenEmbedded build environment for the build host and will not overwrite the customized conf files.

$ source oe-init-build-env

Note: Use bitbake in the build directory.

### Run the Build

core-image-sato

Change directory to the build directory to use bitbake.

To build the core-image, run the command:

$bitbake core-image-sato

Note: Depending on your workstation specifications and internet connect, this could take awhile.

Build Errors

If you have build errors due to package download failure, clean the failed package and rerun the build.

$bitbake -c cleanstate.file

$bitbake file

Resume the full or long build

$bitbake core-image-sato

The cleansstate command works for image recipes also.

### Copy to the SD card

The core or long build can take some time, depending on your internet speed and hardware. There is a difference when copying a long build or a minimal build to the SD card, as well.

Partition the SD card

Partition the SD card with at least 2 partitions using gparted, fdisk, or another tool.

$ sudo apt-get install gparted

Insert the SD card into the workstation.

$ sudo gparted

Use lsblk or df -h to find the microSD card.

Warning: This will format any disk on your workstation. Select the specified SD card listed when using lsblk or df -h. In some cases, the SD card may be mmcblk0 or sdb. Double check to make sure the correct device is being formatted.

Partition 1:

Set the “boot” flag for the first partition.

* type: FAT32
* size: around 30MB
* label: BOOT
* flags: boot and lba

Partition2:

* type: ext4
* size: around 200MB, or remaining SD-card space
* label: ROOT

After partitioning is completed, we are ready to copy the files to the SD-card.

### Long Build

After completing the build, the following images are the most important found in the <TMPDIR>/deploy/images/beaglebone:

bootloader kernel rootfs

* MLO-beaglebone

The second stage bootloader (the first stage bootloader is implemented in ROM code on the AM335x chip and cannot be altered in software)

* u-boot-beaglebone.img

The third stage u-boot bootloader (the “main” bootloader)

Copy the boot loaders. The media or SD may be something like /media/bbb/BOOT/MLO. Check your system for the location to copy to the SD.

$ sudo cp -v MLO-beaglebone /media/bbb/BOOT/MLO

$ sudo cp -v u-boot-beaglebone.img /media/bbb/BOOT/u-boot.img

* zImage

The Linux kernel image (zImage is a special format used with u-boot bootloader) zImage is found in root/boot/

* core-image-base-begalebone.tar.bz2 or something similar

This archive contains the root file system.

$ sudo tar x -C /media/root -f core-image-sato-beaglebone.tar.bz2 /media/bbb/root

After completing the copying process, unmount the SD partitions. Then, insert the micro SD card into the Beaglebone. and continue to the section below on how to boot the Beaglebone.

### \*Minimal RootFS Instructions

If you build a minimal rootfs:

* Install modules-beaglebone.tgz.

$ sudo tar x -C /media/root -f modules-beaglebone.tgz /media/bbb/root

* If you used core-image-base, rootfs already includes the kernel, modules and Device tree files needed to work with the u-boot default configuration
* Install the kernel uImage to /boot in rootfs and install the device tree files into rootfs /boot. Include zImage-4.4-r4am335x-boneblack-xxx.dtb

$ sudo cp uImage-beaglebone.bin /media/root/boot/uImage

$ sudo cp uImage-am335x-bone.dtb /media/root/boot/am335x-bone.dtb

$ sudo cp uImage-am335x-boneblack.dtb /media/root/boot/am335x-boneblack.dtb

After completing the copying process, unmount the SD partitions. Then, insert the micro SD card into the Beaglebone. and continue to the section below on how to boot the Beaglebone.

Reference: <https://www.yoctoproject.org/downloads/bsps/daisy16/beaglebone>

### Boot from the SD card

Connect the microHDMI cable to the Beaglebone Black. Connect the HDMI end to an HDMI output device such as a television or monitor.

The Beaglebone Black has Debian preloaded. You can flash the image to the onboard eMMC so the SD will boot or boot holding down the S2 switch. There are 4 different ways of booting up: eMMC boot, SD boot, Serial boot, and USB boot.

For SD boot, hold the S2 switch on the Beaglebone Black for at least 5 seconds. The lights on the Beaglebone will indicate the BBB is booting.

Login

To login, use the keypad and type the username at the prompt.

Username: root

Password:

The Yocto default username is root. By default, there is no password. After entering root as the username, select the enter key to login.

### cp u-boot.img /media/systebui/BOOT/

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ cp MLO /media/systebui/BOOT/

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp -v MLO-beaglebone /media/systebui/BOOT/MLO

[sudo] password for systebui:

Sorry, try again.

[sudo] password for systebui:

'MLO-beaglebone' -> '/media/systebui/BOOT/MLO'

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp -v u-boot-beaglebone.img /media/systebui/BOOT/u-boot.img

'u-boot-beaglebone.img' -> '/media/systebui/BOOT/u-boot.img'

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-sato-beaglebone.tar.bz2 /media/systebui/Root

tar: core-image-sato-beaglebone.tar.bz2: Cannot open: No such file or directory

tar: Error is not recoverable: exiting now

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-sato-beaglebone.tar.bz2 /media/systebui/Root/tar: core-image-sato-beaglebone.tar.bz2: Cannot open: No such file or directory

tar: Error is not recoverable: exiting now

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 /media/systebui/Root/

tar: /media/systebui/Root: Not found in archive

tar: Exiting with failure status due to previous errors

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 /media/systebui/Root

tar: /media/systebui/Root: Not found in archive

tar: Exiting with failure status due to previous errors

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar -xf core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 -C /media/systebui/Root/

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar -xf modules-beaglebone.tgz -C /media/systebui/Root/

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ $ sudo cp uImage-beaglebone.bin /media/systebui/BOOT/uImage

$: command not found

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ $ sudo cp zImage-beaglebone.bin /media/systebui/BOOT/uImage

$: command not found

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-beaglebone.bin /media/systebui/BOOT/uImage

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage /media/root/boot/am335x-bone.dtb

zImage

zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-am335x-bone-20171205130405.dtb

zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-am335x-boneblack-20171205130405.dtb

zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-beaglebone-20171205130405.bin

zImage-am335x-boneblack.dtb

zImage-am335x-bone.dtb

zImage-beaglebone.bin

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-am335x-bone.dtb /media/systebui/BOOT/am335x-bone.dtb

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-am335x-boneblack.dtb /media/systebui/BOOT/am335x-boneblack.dtb

systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sync

<https://layers.openembedded.org/layerindex/branch/master/layers/>

<http://free-electrons.com/doc/training/yocto/yocto-labs.pdf>

<https://www.yoctoproject.org/downloads/bsps/daisy16/beaglebone>

<http://www.yoctoproject.org/bulk/devday-eu-2014/ypdd14-hallinan-intro-lab.pdf>

[https://www](https://www/)[.org/downloads/bsps/jethro20/beaglebone](https://www.yoctoproject.org/downloads/bsps/jethro20/beaglebone)

<http://www.jumpnowtek.com/beaglebone/BeagleBone-Systems-with-Yocto.html>

<https://www.hackster.io/30847/yocto-on-beaglebone-black-9ae649>

# U-Boot with fast cp u-boot.img /media/systebui/BOOT/

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ cp MLO /media/systebui/BOOT/

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp -v MLO-beaglebone /media/systebui/BOOT/MLO

# [sudo] password for systebui:

# Sorry, try again.

# [sudo] password for systebui:

# 'MLO-beaglebone' -> '/media/systebui/BOOT/MLO'

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp -v u-boot-beaglebone.img /media/systebui/BOOT/u-boot.img

# 'u-boot-beaglebone.img' -> '/media/systebui/BOOT/u-boot.img'

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-sato-beaglebone.tar.bz2 /media/systebui/Root

# tar: core-image-sato-beaglebone.tar.bz2: Cannot open: No such file or directory

# tar: Error is not recoverable: exiting now

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-sato-beaglebone.tar.bz2 /media/systebui/Root/tar: core-image-sato-beaglebone.tar.bz2: Cannot open: No such file or directory

# tar: Error is not recoverable: exiting now

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 /media/systebui/Root/

# tar: /media/systebui/Root: Not found in archive

# tar: Exiting with failure status due to previous errors

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar x -C /media/systebui/Root/ -f core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 /media/systebui/Root

# tar: /media/systebui/Root: Not found in archive

# tar: Exiting with failure status due to previous errors

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar -xf core-image-base-beaglebone-20171205130405.rootfs.tar.bz2 -C /media/systebui/Root/

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo tar -xf modules-beaglebone.tgz -C /media/systebui/Root/

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ $ sudo cp uImage-beaglebone.bin /media/systebui/BOOT/uImage

# $: command not found

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ $ sudo cp zImage-beaglebone.bin /media/systebui/BOOT/uImage

# $: command not found

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-beaglebone.bin /media/systebui/BOOT/uImage

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage /media/root/boot/am335x-bone.dtb

# zImage

# zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-am335x-bone-20171205130405.dtb

# zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-am335x-boneblack-20171205130405.dtb

# zImage--4.1.15+git0+46bb64d605\_efb6ffb2ca-r0-beaglebone-20171205130405.bin

# zImage-am335x-boneblack.dtb

# zImage-am335x-bone.dtb

# zImage-beaglebone.bin

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-am335x-bone.dtb /media/systebui/BOOT/am335x-bone.dtb

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sudo cp zImage-am335x-boneblack.dtb /media/systebui/BOOT/am335x-boneblack.dtb

# systebui@VTLW046L:~/build/tmp/deploy/images/beaglebone$ sync

# boot for the BeagleBone Black

This is a fork of U-Boot which adds support for the Android fastboot protocol. The changes are somewhat specific to the BeagleBone Black.

# Build from source

Clone this repository:

$ git clone https://github.com/csimmonds/u-boot.git

$ cd u-boot

$ git checkout am335x-v2013.01.01-bbb-fb

Since you are likely to be building this along with AOSP, I suggest using the Android cross compiler from prebuilts, but probably any recent arm eabi toolchain will do. If you have sourced build/envsetup.sh and selected the lunch combo the path will be set up already. If not, set it now, substituting ${AOSP} with the place where you installed your AOSP. I used Android 4.4, which has gcc v4.7:

$ PATH=${AOSP}/prebuilts/gcc/linux-x86/arm/arm-eabi-4.7/bin:$PATH

Then configure and build U-Boot:

$ make CROSS\_COMPILE=arm-eabi- distclean

$ make CROSS\_COMPILE=arm-eabi- am335x\_evm\_config

$ make CROSS\_COMPILE=arm-eabi-

This produces the two files: MLO and u-boot.img

# Create a bootable micro SD card

Take a micro-SD card and connect it to your PC, either using a direct SD slot if available, in which case the card will appear as "/dev/mmcblk0" or, using a memory card reader in which case the card wil be seen as "/dev/sdb", or "/dev/sdc", etc

Now type the command below to partition the micro-SD card, assuming that the card is seen as "/dev/mmcblk0"

sudo sfdisk -D -H 255 -S 63 /dev/mmcblk0 << EOF

,9,0x0C,\*

,,,-

EOF

Format the first partition as FAT32

sudo mkfs.vfat -F 32 -n boot /dev/mmcblk0p1

Remove the card and insert it again. It should automatically be mounted as "/media/boot".

Now, copy the files to this partition:

cp MLO u-boot.img /media/boot

Finally, umount it.

# Flash U-Boot to the BeagleBone Black

You will need

* A BeagelBone Black rev A/B/C
* A micro SD card of any capacity since you are only going to use the first 70 MiB to write a small flasher image
* The mini USB to USB A cable supplied with the BeagleBone
* A 5V power supply because the current used when writing to the eMMC chip may exceed that supplied by a typical USB port.

The procedure is:

1. With no power on the BeagleBone, insert the microSD card
2. Press and hold the 'Boot button' on the BeagleBone, power up the board using the external 5V power connector and release the button after the fastboot LED (USER 0 led) lights up, showing that it is running this version of U-Boot and is ready to accept fastboot commands
3. Plug in the USB cable between the mini USB port on the BeagleBone and the PC. Then, using the fastboot command from the Android SDK or an AOSP build, check that the BeagleBone has been detected by typing (on the PC)

$ fastboot devices

90:59:af:5e:94:81 fastboot

1. If instead you see

$ fastboot devices

no permissions fastboot

Add this line to /etc/udev/rules.d/51-android.rules

SUBSYSTEM =="usb", ATTRS{idVendor}=="0451", ATTRS{idProduct}=="d022" , MODE="0666"

Then unplug the mini USB cable and plug it back in again. Check that the permissions problem has gone away.

1. Use fastboot to format the eMMC chip and then flash the images

$ fastboot oem format

$ fastboot flash spl MLO

$ fastboot flash bootloader u-boot.img

1. Power off the board and remove the SD card
2. Power on again. Your BeagleBone will boot into U-Boot and it will be once again ready to accept fastboot commands

<https://github.com/csimmonds/u-boot>