# Patching Xenomai on emdebian for Beagle Bone Black on Ubuntu 13.10(Saucy)

The guide is heavily based on

http://github.com/DrunkenInfant/beaglebone-xenomai/blob/master/beaglebone-xenomai-debian.md

Installing xenomai and debian on the beagle bone black by SagarBehere[?](https://code.google.com/p/rtes-group-7/w/edit/SagarBehere)

## Steps you need to follow

On Ubuntu 13.10(Saucy) some of the packages for tool- chains and image making are missing

**First install the following packages**

sudo apt-get install gcc-arm-linux-gnueabi

sudo apt-get install gcc-arm-linux-gnueabihf

sudo apt-get install u-boot-tools

If something fails try installing below as well:

apt-get install g++-4.7-arm-linux-gnueabihf gcc-4.7-arm-linux-gnueabihf

Creating sym-links for the binaries:

ln -s /usr/bin/arm-linux-gnueabihf-gcc-4.7 /usr/bin/arm-linux-gnueabi-gcc  
ln -s /usr/bin/arm-linux-gnueabihf-g++-4.7 /usr/bin/arm-linux-gnueabi-g++

### Obtain and prepare all sources

In this section, we will clone various repositories and checkout their correct branches/commits. Then we'll apply patches to and from various software sources. For the purpose of this document, all the repositories will be cloned into $HOME/bbb and all subsequent work will be done in this directory.

mkdir ~/bbb cd ~/bbb 1.2.1 kernel sources

kernel sources from the xenomai project's ipipe-jki git repository. We will be using the branch corresponding to the 3.8.13 kernel, which is called for-upstream/3.8

git clone git://git.xenomai.org/ipipe-jki.git  
cd ipipe-jki;   
git checkout -b 3.8 origin/for-upstream/3.8

#### Xenomai sources

git clone git://git.xenomai.org/xenomai-2.6.git

#### Beaglebone-xenomai scripts

Contains general beaglebone specific patches

git clone https://github.com/DrunkenInfant/beaglebone-xenomai

#### Meta-beagleboard patches

Some more beaglebone specific patches to the kernel sources

git clone https://github.com/beagleboard/meta-beagleboard.git  
cd meta-beagleboard

We need to check out the particular commit corresponding to kernel 3.8.13. We can find this commit by examining the output of the command

git log -S3.8.13 --source --all

which indicates that commit with hash 50316366dd4f75027ee5291b65a9bbcfa9a9e840is the one we need.

git checkout 50316366dd4f75027ee5291b65a9bbcfa9a9e840

**WARNING!** All patches from this repository do not apply cleanly due to some (very minor) issues. Applying the meta-beaglebone-xenomai.patch from the beaglebone-xenomai repository fixes this.

**run and install all three patches next**

patch -p1 < ~/bbb/beaglebone-xenomai/meta-beaglebone-xenomai.patch

(**NOTE**: the meta-beaglebone-xenomai.patch contains patches for three files, one of which was missing from my meta-beagleboard repository as prepared so far)

#### Apply meta-beagleboard patches to kernel sources

Open the file: ~/bbb/beaglebone-xenomai/apply-beaglebone-patches.sh

and change the META\_BEAGLEBONE\_ROOT to the location of your cloned version of the meta-beaglebone repo. If you have followed the steps above, the relevant line should look like

META\_BEAGLEBONE\_ROOT=~/bbb/meta-beagleboard

**CAUTION!**: Verify that the PATCHSETS variable in the same file does not contain any directories that are not present in the meta-beagleboard's common-'bsp/recipes-kernel/linux/linux-mainline-3.8' directory. If the PATCHSETS variable has extra directories, remove them.

cd ~/bbb/ipipe-jki

try if this works(command from original guide):

source ~/bbb/beaglebone-xenomai/apply-beaglebone-patches.sh

I had to manually change my directory to do this 'sudo' will be required in most of the commands

These patches SHOULD apply cleanly.

### Prepare the kernel sources for xenomai

cd ~/bbb/ipipe-jki

~/bbb/xenomai-2.6/scripts/prepare-kernel.sh --arch=arm

Should be smooth up to this point. Now starts the messy part.

**Configuring the Xenomai**

The below command fails if you do not have dependency `libncurses'

you may try `apt-get sudo libncurses5 OR lib64ncurses5 OR b64ncurses5-dev OR lib64tinfo5 '

**WARNING** Change your terminal window to fullscreen before executing the below command

make ARCH=arm menuconfig

Configure the patch as per your requirement: (This is the part we need to discuss. What should be included in the debian build)

There are some Drivers that fails the next make command : Things that require to be removed which are pre-selected:

1. SCSI Driver support has to be removed
2. All the Accelerometer Drivers have to be removed Industrial I/O Support is the menu for this

Another thing that will cause the next make command to fail:

<https://drive.google.com/file/d/0B_A7pt_CBFkCUnFiN0tpR04wYTQ/edit?usp=sharing>

Here the of\_free\_overlay\_info which is a function inside the has been defined twice in the file ~/bbb/ipipe-jkl-/include/linux/of.hin a #ifdef fashion. To negate this is just sudo gedit the file and delete the second declaration for it to work.

Minimum Steps for configuration:

1. Make sure the submenu 'Real-time sub-system' exists, if not the prepare-kernel script was not successful.
2. Disable CPU frequency scaling, 'CPU Power Management ---> CPU Frequency scaling'
3. Check the Real-time sub-system menu. If there are any conflicts left there will be a warning in that menu.

#### The most tiresome part is next

This will create a image from the configuration you just created

This creates images for all Xenomai supported hardware, Will try to figure out how to stop it from doing it for Blackfin , x86, etc as we need it only for ARM.

make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- ZRELADDR=0x80008000 uImage modules

This may fail saying that multiple instance of some function is declared in some c file, i don't recall as after compile the directories before generation of image are removed after.

If this fails midway , it is because of some drivers you requested to be configured so do

make ARCH=arm menuconfig

again and remove the driver which is causing the image build process to fail.

Once this is done you can find the image in multiple file formats like uImage and zImage files under the directory ~/bbb/ipipe-jki/arch/arm/boot/

### Compile xenomai userspace

cd ~/bbb/xenomai-2.6  
./configure --host=arm-linux-gnueabihf  
make

This is as far as i have tested so far, but the consequent steps require formatting the SD card to a provided format:

### Preparing the SD Card with rootfs

git clone git://github.com/RobertCNelson/netinstall.git  
sudo ./mk\_mmc.sh --mmc /dev/sdX --dtb am335x-boneblack --distro wheezy-armhf --firmware

Make sure to substitute /dev/sdX in the command above with the correct device for your sd card.

Alternatively, you can download and use the ready image as described at

http://elinux.org/BeagleBoardDebian#Demo\_Image

Now we need to copy the newly compiled kernel to this sd card.

mount /dev/sdX1 /mnt  
cp /mnt/zImage /mnt/zImage-original  
cp ~/bbb/ipipe-jki/arch/arm/boot/zImage /mnt/  
sync  
umount /mnt

Finally, copy the kernel modules and firmware to the rootfs

mount /dev/sdX2 /mnt  
cd ~/bbb/ipipe-jki  
make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- modules\_install INSTALL\_MOD\_PATH=/mnt/  
make ARCH=arm CROSS\_COMPILE=arm-linux-gnueabihf- firmware\_install INSTALL\_FW\_PATH=/mnt/  
sync  
umount /mnt

And that's it! Insert the sd card into your beaglebone/black and it should boot with the xenomai kernel 3.8.13 and debian wheezy userland.

Once the board boots, you may find it necessary to perform the following

addgroup xenomai  
echo '/usr/xenomai/lib' >> /etc/ld.so.conf  
ldconfig

Test whether xenomai works by doing, for example, the latency test

cd /usr/xenomai/bin  
./latency -p0 -t1