Workshop Setup Guide

Introduction

This document contains information for setting up an Ubuntu 12.04 host computer to run the lab exercises of the “Introduction to Embedded Linux One-Day Workshop.”

It consists of 4 required sections:

- Installing Ubuntu 12.04
- Installing Code Composer Studio
- Installing Lab Files
- Configuring Ubuntu Static IP

After completing these installation steps, you will have everything needed to run the lab exercises on your system.

Additionally, a number of steps were taken to make the environment more user friendly (“Installing Gnome3 and Standard Scrollbars”) and to set up the lab files toolchain and target filesystem. These comprise the three optional sections. There is no need to go through the optional sections in order to run the lab exercises, but if you would like to know the steps that were required to set up that portion of the lab environment, the steps are shown in these optional sections.

The lab files that you will need to install are located on the workshop wiki page at:

http://processors.wiki.ti.com/index.php/Introduction_to_Linux_One-Day_Workshop

Chapter Topics

Installing Ubuntu 12.04

There are many tutorials available for installing Ubuntu, so this section will not go through great detail on the actual installation; however, it provides an optional section for removing the user password and setting automatic login, as is done in the workshop image.

1. **Begin by downloading an Ubuntu 12.04 image and burning onto an installation disk.**
2. **Install Ubuntu 12.04 on your computer.**
   Other versions of Ubuntu may also work, but version 12.04 is what has been tested for this workshop.
   If you select “automatic login” on the user setup screen, you can skip step 3.
   Be sure to write down the password that you set! The following steps will show you how to remove the password, but you will need to know the old one.
3. **Open a terminal**
   ctrl-alt-t
4. **Select automatic login (If you forgot to select in step 2)**
   `# sudo gedit /etc/lightdm/lightdm.conf`
   Enter the password from step 2 when prompted.
   Add the following four lines under the section header “[SeatDefaults]”
   ```
   autologin-guest=false
   autologin-user=user
   autologin-user-timeout=0
   autologin-session=lightdm-autologin
   ```
5. **Allow null passwords for sudo**
   `# sudo gedit /etc/sudoers`
   Enter the password from step 2 if prompted.
   Locate the line that reads:
   `%admin ALL=(ALL) ALL`
   And change to read
   `%admin ALL=(ALL) NOPASSWD: ALL`
6. **Allow null passwords for authorization (i.e. login)**
   `# sudo gedit /etc/pam.d/common-auth`
   Locate the line that contains “nullok_secure” and change “nullok_secure” into just “nullok”
7. **Remove user password**
   `# sudo passwd -d user`
8. **Reboot to test**
   ```
   # sudo shutdown -h now
   When Ubuntu reboots, open a terminal and try the sudo command:
   # sudo ls
   ```
   If everything has worked correctly, the list operation should complete without prompting for a password.

Installing Code Composer Studio v.5.3.0.00090

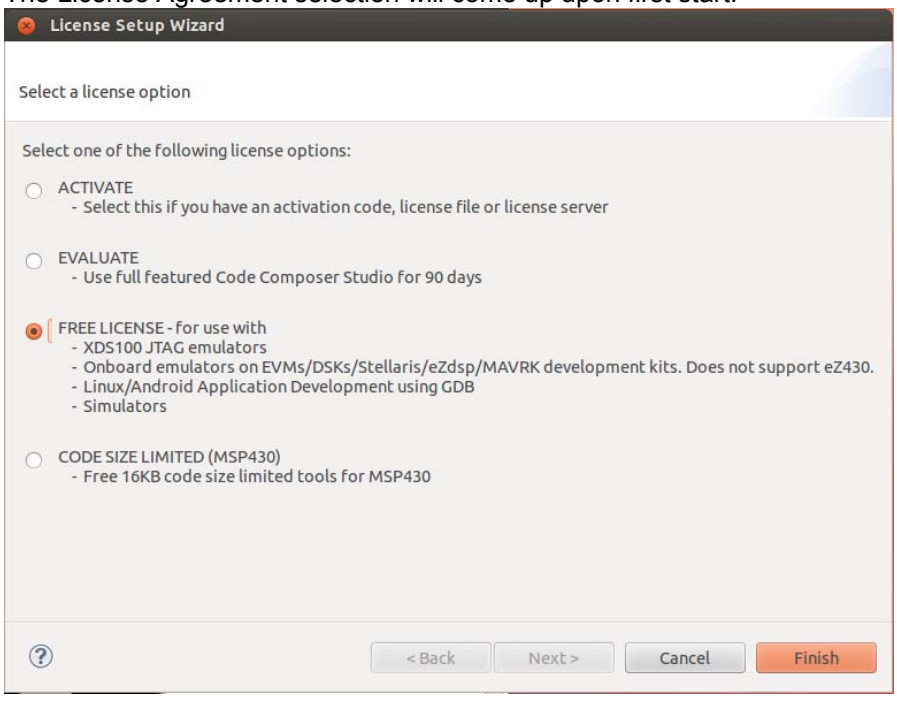
You can download CCS from:

`http://processors.wiki.ti.com/index.php/Download_CCS`

and selecting the “Linux” download button.

Also, because older versions of CCS are not always archived, the exact version used in the workshop is also available from the wiki page.

9. **Download CCS from the above listed link**
10. **Add executable permission to the installer**
    `# chmod a+x ccs_setup_5.3.0.00090.bin`
11. **Run the installer program**
    `# ./ccs_setup_5.3.0.00090.bin`
    Note: do not run the installer with root (i.e. sudo) permissions.
    The installer will give you a message that Emulation drivers can only be installed with root permissions, but we will not be using emulation drivers, so this is not an issue.
12. **Read and accept the license agreement.**
13. **Accept the default install directory: /home/user/ti**
14. **Choose custom install**
15. **Select “AMxx Cortex-A and ARM9 processor” from the Processor Support window**
16. **In “Select Components” window, select nothing.**
17. **Deselect all emulator support except “TI Emulators” on the next screen**
    The workshop does not use “TI Emulators” but there is no way to deselect in the installer.
18. **Press “Next” on the next two screens to begin the installation**
19. **Select “Create Desktop Shortcut” on final screen (Should be selected by default.)**
20. **Start Code Composer Studio**
21. **Select the Free License Agreement**
    The License Agreement selection will come up upon first start.



If for some reason the License Setup Wizard does not automatically launch, you can access it via:

Help→Code Composer Studio Licenscing Information→Upgrade Tab→Launch Licensing Setup…

22. **Close CCS**

Installing Lab Files

Note that the Code Composer Studio project files that have been set up for the workshop expect for the labs folder to be extracted into the "/home/user" directory. If you have a previous Ubuntu installation with a different user name, it is recommended that you create a new user named "user" and install the lab files into this user's directory. You should then run the lab exercises when logged in as "user."

23. **Update the Aptitude package manager**
    ubuntu$ sudo apt-get update
24. **Install the sg3-utils package**
    ubuntu$ sudo apt-get install sg3-utils
25. **Install python 2.6 (used by arm-arago-linux-gnueabi-gdb)**
    ubuntu$ sudo add-apt-repository ppa:fkrull/deadsnakes
    ubuntu$ sudo apt-get update
    ubuntu$ sudo apt-get install python2.6 python2.6-dev
26. **Download lab files**
    ubuntu$ firefox
    browse to:
    http://processors.wiki.ti.com/index.php/Introduction_to_Linux_One-Day_Workshop
    Download "bbb_linux_oneday_labs.tar.gz" and "Angstrom_host_tools.tar.gz"
27. **Install lab files**
    ```
    ubuntu$ tar -zxf bbb_linux_oneday_labs.tar.gz –C /home/user
    ```
28. **Install Angstrom toolchain**
    ```
    ubuntu$ cd /home/user/labs/lab00_toolchain
    ubuntu$ sudo ./angstrom-eglibc-i686-armv7a-vfp-neon-v2012.12-
    toolchain.sh
    ```

    NOTE: be certain to accept the default installation location, "/usr/local/oecore-i686"

    The location of standard header files and libraries is hard coded into the gcc and g++ executables. If you relocate them to a location other than the default as listed above, they will not work correctly, specifically they will not be able to locate libraries without the addition of a "—sysroot=<dir>" directive.
29. **Create tools link**
    Open Embedded/Yocto is of necessity very organized, and reaches this organization through the use of many layers of subdirectories. This makes the cross compiler location a little difficult to find.
    The cross-compiler build tools are not relocatable because the include headers and libraries need to be in a specific location for the compiler to find them. A good solution then is to create a soft link to the tools folder at a location that is easy to find, the user's home directory.

    ```
    ubuntu$ sudo ln –s /usr/local/oecore-686/i686-angstromsdk-
    linux/usr/bin/armv7a-vfp-neon-angstrom-linux-gnueabi
    /home/user/armv7a-vfp-neon-angstrom-linux-gnueabi
    ```

    Note that this is all on one line and has the following two components:
    `/usr/local/oecore-686/i686-angstromsdk-linux/usr/bin/armv7a-vfp-neon-angstrom-linux-gnueabi`
    Separated by a space from
    `/home/user/armv7a-vfp-neon-angstrom-linux-gnueabi`

    The code composer studio files that have been provided reference /home/user/armv7a-vfp-neon-angstrom-linux-gnueabi, so if this link is left off, the projects will not build.

30. **Copy the gnu debugger to the toolchain directory**
    ```
    ubuntu$ cp /home/user/labs/lab00_toolchain/arm-arago-linux-
    gnueabi-gdb /home/user/armv7a-vfp-neon-angstrom-linux-gnueabi
    ```
31. **Copy .gdbinit to home directory**
    ```
    ubuntu$ cp /home/user/labs/lab00_toolchain/gdbinit
    /home/user/.gdbinit
    ```

    Note that the destination must be ".gdbinit" which is preceded with a period. The name was used without a prepended period in the toolchain directory because files that begin with a period are hidden.

Configuring Ubuntu Static IP

The “auto” setting for usb0 in /etc/network/interfaces is a workaround. It would be better specified as “allow-hotplug” however, there are known issues with this in Ubuntu 12.04. The web recommends using udev as an alternate solution, but the workshop developer was unable to make this approach work.

Using “auto usb0” works well, but with the disadvantage that if no ethernet-over-usb connection is available when Ubuntu starts up, the message “waiting on network configuration…” will appear and will require about 2 minutes to timeout. This extra 2 minutes of boot time may be circumvented by attaching the beaglebone so that the interface is present.

Users who dislike this 2 minute boot time may remove “auto usb0” in which case the usb0 will have to be manually configured each time the Beaglebone is attached using “#sudo ifup usb0”

32. **Open /etc/network/interfaces file**
    # sudo gedit /etc/network/interfaces
33. **Add an “eth0” entry (or modify current entry.) Entry should be as follows:**

```
auto eth0
iface eth0 inet static
      address 192.168.1.1
      netmask 255.255.255.0

auto usb0
iface usb0 inet static
      address 192.168.2.1
      netmask 255.255.255.0
```

    Note: “address” and “netmask” entries preceded by tab.
34. **Save and save and close**
35. **Remove Gnome networking settings**

```
# sudo nm-connection-editor
```

    Any connection that appears under the “wired” or “wireless” tab should be deleted.
36. **(Optional) Reboot and use “ifconfig” to verify new setting**

```
# ifconfig
```

37. **Open /etc/hosts**

```
# sudo gedit /etc/hosts
```

38. **Add static IP addresses for hosts on the network**
    (At the end of the file, add the following)

```
192.168.1.1 ubuntu.gigether.net
192.168.1.2 beaglebone.gigether.net
192.168.2.1 ubuntu.etherusb.net
192.168.2.2 beaglebone.etherusb.net
```

(Optional) Installing Gnome3 and Standard Scrollbars

Ubuntu 12.04 ships with a desktop manager called Unity. One feature that a lot of people do not prefer in Unity is that the drop-down lists that would normally appear at the top of a window (including CCS) now appear at the top of the desktop. Additionally, Unity uses a new type of scrollbar called overlay scrollbars that, while saving a little space on the screen that can be used for other things, are a little more difficult to use.

This section is not required for the workshop labs to work properly, but since these changes were made on the workshop image, they are listed here.

39. **Launch a terminal**
40. **Acquire a WAN (i.e. internet) connection**
    If you have already set up a static IP address as per the previous section, you can override the static address using
    ```
    ubuntu$ sudo ifdown eth0
    ubuntu$ sudo dhclient eth0
    ```
41. **Install gnome-shell Aptitude package**
    ```
    ubuntu$ sudo add-apt-repository ppa:gnome3-team/gnome3
    ubuntu$ sudo apt-get update
    ubuntu$ sudo apt-get install gnome-shell
    ```
42. **Log out of the Ubuntu session**
    There is a gear icon in the top right corner that produces a drop-down menu with the logout option.
43. **Select the Gnome Desktop**
    Click the Ubuntu icon next to thye username (user) and select Gnome.
44. **Press login to log back in**
    The desktop has only subtly changed, but if you launch CCS, you will notice that the pulldown menus are now at the top of the CCS window (instead of along the top of the desktop.)
45. **Disable overylay scrollbars**
    Launch a terminal and type the following (single line, no carriage return)
    ```
    ubuntu$ gsettings set org.gnome.desktop.interface ubuntu-overlay-scrollbars false
    ```
46. **Log out and back in for change to take effect**
    The “user” dropdown menu in the top right of the desktop can be used to log out.

(Optional) Installing Angstrom Cross-compile Tools

NOTE: All of the cross-compile tools built in this section are included in the workshop lab installation file, and do not need to be rebuilt in order to run the workshop. These steps are included here as a reference for those who wish to know what was done.

The Angstrom distribution that is used in the workshop provides a set of cross-compile tools. These can be rebuilt using the bitbake build system of OpenEmbedded.

Note that, at the time of the creation of this document, the version of gdb built in step 54 is incompatible with the version of eclipse used in Code Composer Studio due to the deprication of the "python printk-stack off" command as per this link:

http://stackoverflow.com/questions/14950138/eclipse-cdt-gdb-error-undefined-maintenance-set-command-python-print-stack

While it may be possible to upgrade the eclipse support in CCS as suggested in the article, the decision was made to instead use the cross-gdb found in the AM335x evm software development kit, which is based on an earlier gdb version before this command was depricated.

Hence the lab installation file actually uses "arm-arago-linux-gnueabi-gdb" instead of "arm-angstrom-linux-gnueabi-gdb." The build procedure is what was used to procure the build tools (gcc, g++) used in the labs.

47. **Acquire a WAN (i.e. internet) connection**
    If you have already set up a static IP address as per the previous section, you can override the static address using
    ```
    ubuntu$ sudo ifdown eth0
    ubuntu$ sudo dhclient eth0
    ```
48. **Install Bitbake/OE dependencies**
    ```
    ubuntu$ sudo apt-get update
    ubuntu$ sudo apt-get install subversion cvs git-core \
    build-essential help2man diffstat texi2html texinfo \
    libncurses5-dev gawk python-dev python-pysqlite2 \
    gnome-doc-utils gettext automake flex chrpath
    ```
49. **Install git**
    ```
    ubuntu$ sudo apt-get install git
    ```
50. **"git" Angstrom install scripts**
    ```
    ubuntu$ cd ~
    ubuntu$ mkdir Angstrom
    ubuntu$ cd Angstrom
    ubuntu$ git clone git://github.com/Angstrom-distribution/setup-scripts.git
    ```
51. **Link /bin/sh into /bin/bash (currently it is /bin/dash)**
    The OpenEmbedded build system requires /bin/bash
    ```
    ubuntu$ sudo rm /bin/sh
    ubuntu$ sudo ln –s /bin/bash /bin/sh
    ```
52. **Configure OpenEmbedded for beaglebone**
    ```
    ubuntu$ cd setup-scripts
    ubuntu$ MACHINE=beaglebone ./oebb.sh config beaglebone
    ```
53. **Build the cross-compiler toolchain**
    ```
    ubuntu$ MACHINE=beaglebone ./oebb.sh bitbake meta-toolchain
    ```
54. **Build cross gdb (the gnu debugger), which also builds gdbserver**
    ```
    ubuntu$ MACHINE=beaglebone ./oebb.sh bitbake gdb-cross
    ```

55. **(Optional) Extract Tools**
    Located at:
    `setup-scripts/build/tmp-angstrom_v2012_12-eglibc/deploy/sdk`
    Important that when you extract, you use the default location of /usr/local/oecore-i686 as this is where the search paths are hard coded to.
56. **Insert Beaglebone Black micro-SD card into cardreader and host machine**
57. **Copy Angstrom sysroot to Beaglebone micro-SD card**
    ```
    ubuntu$ sudo cp –R /home/user/Angstrom/setup-scripts/build/tmp-angstrom_v2012_12-eglibc/sysroots/beaglebone/* /media/Angstrom
    ```
58. **Copy gdbserver to Beaglebone micro-SD card**
    ```
    ubuntu$ sudo cp /home/usr/Angstrom/setup-scripts/build/tmp-angstrom_v2012_12-eglibc/work/armv7a-vfp-neon-angstrom-linux-gnueabi/gdb-7.5-r0.0/package/usr/bin/gdbserver /media/Angstrom/usr/bin/gdbserver
    ```
59. **Eject micro-SD**
    ```
    ubuntu$ sudo eject /media/Angstrom
    ```

(Optional) Modifying Angstrom Filesystem

NOTE: All of the filesystem modifications listed in this section have already been made to the filesystem image included in the lab exercises installation file and do not need to be rebuilt in order to run the workshop. These steps are included here as a reference for those who wish to know what was done.

By default the Angstrom Filesystem used in this workshop uses DHCP to acquire an IP address. This section demonstrates using the "connman" (connection manager) utility from Angstrom in order to set a static IP address.

The ethernet over usb gadget driver cannot be configured using connman, so a systemd startup script is created to launch this driver at each startup. (Angstrom uses systemd instead of sysV as its startup scripting.)

The starting micro-SD card image is:

Angstrom-Cloud9-IDE-GNOME-eglibc-ipk-v2012.12-beaglebone-2013.06.17.img.xz

Which was downloaded from:

http://downloads.angstrom-distribution.org/demo/beaglebone/

The finished filesystem is provided on the workshop wiki page, but the steps are also listed here.

60. **Boot the Beaglebone Black attached to a router that provides access to the wide area network**
61. **Browse to the router, log in and determine the IP address of the Beaglebone Black from the DHCP list of the router.**
    If you are unsure of the IP address of the router, it is usually either 192.168.1.1 or 10.0.0.1
    If neither of these works, attach an x86 PC, acquire a DHCP address from the router, and check the gateway address in your IP settings.
62. **Force Ubuntu to acquire a DHCP address from the router**
    This is necessary assuming that you have set the Ubuntu IP address statically as per the previous section.
    ```
    ubuntu# sudo ifdown eth0
    ubuntu# sudo dhclient eth0
    ```
63. **From Ubuntu, create a secure shell connection to the Beaglebone Black**
    ```
    ubuntu# ssh root@192.168.1.2
    ```
    You may be told that there is no ssh key for the connection and asked if you would like to create one, to which say "yes."
    When you log into the Beaglebone Black, there is no password (press enter when prompted.)
64. **On the Beaglebone Black, update the opkg package manager**
    ```
    bbb# opkg update
    ```
65. **Install connman-tests package**
    ```
    bbb# opkg install connman-tests
    ```
66. **Change to the connman test script directory**
    ```
    bbb# cd /usr/lib/connman/test
    ```
67. **List the currently configured connections**
    ```
    bbb# ./get-services
    ```
    The first line of the output should list something similar to:
    ```
    [ /net/connman/service/ethernet_405fc276b749_cable ]
    ```
    (The hexadecimal hash following "ethernet_" will probably be different on your system.)

68. **Use set-ipv4-method script to set a static IP address using the connection name identified in step 67**
    ```
    bbb# ./set-ipv4-method ethernet_405fc276b749_cable manual
    192.168.1.2 255.255.255.0
    ```
    (note: above is single line with no carriage return.)
69. **Attach the Beaglebone Black to the Ubuntu host using an ethernet crossover cable, reset both**
    Note that the network cards of most modern computers will detect a crossover configuration and automatically switch even with a standard ethernet cable.
70. **Use secure shell to create a connection into the Beaglebone Black**
    ubuntu# ssh root@192.168.1.2
71. **Change to /lib/systemd/system**
    ```
    bbb# cd /lib/systemd/system
    ```
72. **Create file "etherusb.service" with vi editor**
    ```
    bbb# vi etherusb.service
    ```
    Note: if you don't want to use vi, you can use the MMC card reader to creae this file on your Ubuntu pc using gedit.
73. **Press "i" to enter insert mode**
74. **Enter the following into the file:**
    ```
    [Unit]
    Description=Turn on usb0

    [Service]
    Type=oneshot
    ExecStart=/lib/systemd/system/etherusb.sh

    [Install]
    WantedBy=multi-user.target
    ```
75. **Press "ESC" key to enter command mode**
76. **Press ":w" to save the file**
77. **Press ":q" to exit**
78. **Create file "etherusb.sh" with vi editor**
    ```
    bbb# vi etherusb.sh
    ```
79. **Press "i" to enter insert mode**
80. **Enter the following into the file:**
    ```
    #!/bin/sh
    /sbin/modprobe g_ether
    sleep 10
    /sbin/ifconfig usb0 192.168.2.2 netmask 255.255.255.0
    ```
81. **Press ":w" to save the file**
82. **Press ":q" to exit**
83. **Make etherusb.sh executable**
    ```
    bbb# chmod a+x etherusb.sh
    ```
84. **Register etherusb.service with systemd**
    ```
    bbb# systemctl enable etherusb.service
    ```
85. **Test the startup service**
    ```
    bbb# systemctl start etherusb.service
    ```
86. **Verify usb0 is set with ifconfig**
    ```
    bbb# ifconfig
    ```
    You should see an entry for usb0 with address 192.168.2.2
87. **(Optional) Reboot Beaglebone Black and verify usb0 with ifconfig**
88. **Edit the file "/etc/hosts" with vi editor**
    ```
    bbb# vi /etc/hosts
    ```
89. **Press "i" to enter insert mode**

90. Add the following lines at the end of the file:

```
192.168.1.1     ubuntu.gigether.net
192.168.1.2     beaglebone.gigether.net
192.168.2.1     ubuntu.etherusb.net
192.168.2.2     beaglebone.etherusb.net
```

91. Press ":w" to save the file

92. Press ":q" to exit