Quick Start Guide for BeagleBone

by Brian Fraser
Last update: Sept 24, 2017

This document guides the user through:

1. Installing Ubuntu in a virtual machine.
2. Connecting to the target using serial port and SSH.
3. Cross compile and run on the target.

Table of Contents

Note: Guide not yet fully tested in SFU Surrey Linux Lab (SUR4080). Some changes may be needed.

Formatting:

1. Commands starting with `$` are Linux console commands on the host PC:
   ```
   $ echo "Hello world!"
   ```
2. Commands starting with `#` are Linux commands on the target (BeagleBone):
   ```
   # echo "Hello embedded world!"
   ```
3. Almost all commands are case sensitive.

Revision History:

- Sept 19: Initial version for Fall 2017
- Sept 24: Updated dmesg output when connecting USB-TTY device for /dev/ttyUSB0

1. Host OS Setup

1. Linux Ubuntu 16.04.xx, 64bit, is the supported OS on the host PC for this class. Other versions of Linux or Unix derivatives may work, but are not supported by the instructor and TA.
   - You can download Ubuntu from its official site: http://www.ubuntu.com/download
     A copy is available via http://mirror.cs.surrey.sfu.ca for access **from on campus**.
2. If you are running a non-Linux OS, you may install Ubuntu in a virtual machine (VM). However, note that running inside a VM means one extra level of configuration is required.
   - Under Windows or Mac OS, you can use VMware Worstation Player or VirtualBox (both free).
     This YouTube playlist features three videos showing how to install a VM.

     I recommend VMware over VirtualBox as I find VirtualBox sometimes messes up which USB devices are mapped to my VM which forces me to reboot the computer.
     - If you use VM Player, some students have found problems if you plug any USB 2.0 hardware into a USB 3.0 slot on your computer. If you are having USB problems, try rebooting your VM and your host OS, and switching the port you are plugging the device into ensuring it is a USB 2.0 device).
   - Directions for in SFU CSIL Labs (SUR 4080, and possibly all other CSIL labs):
     - VM Ware is installed in the labs. When you create a virtual machine it will default to "~/VirtualBox VMs" which is remapped to /usr/shared/CMPT/VirtualBox/<yourID>
       - This directory has more space than your home folder and is shared by all CSIL machines so you'll have access to your VM on any PC in CSIL.
       - The folder is access protected so only you have access to it.
     - Note that you cannot complete the assignments in the lab without a virtual machine because you do not have root permissions on the lab computers. We'll need root permissions to setup file sharing and install/upgrade programs.
     - When creating a virtual machine in the lab, I recommend 4GB ram, 20GB Hard drive, Linux Ubuntu x64.
3. If you install Linux in a VM:
   - Make the virtual hard drive large (>70GB, maybe 100GB). This gives plenty of room for extra build tools, libraries, and such.
     - In the lab, use a smaller hard drive because it is using up shared space.
   - Give it at least 2GB (up to 8GB) of RAM, video RAM, and as many processor as possible; this will speed up Linux. Also enable 3d graphics acceleration for improved performance.
4. Troubleshooting:
   - If having problems with Ubuntu's `apt-get` and dependencies, try following this guide: http://askubuntu.com/questions/140246/how-do-i-resolve-unmet-dependencies-after-adding-a-ppa
     Note that if using the software-properties-gtk tool, you may need to run it with sudo from the command line:
     ```
     $ sudo software-properties-gtk
     ```

1.1 Virtual Machine on USB Stick

It’s possible to install a Virtual Machine onto a USB memory stick so that you can take it with you between computers. If working in the lab, it is recommended you use the CSIL shared space (described above) because it will be faster and store the data more reliably.

- Get a USB 3.0 memory stick! Need 32+ GB likely. Insert it into the “SuperSpeed” USB port on the computers (has an “SS”). Note the top two USB ports on the front of the lab PCs are USB 3.0; the others are 2.0. Best to have your USB stick formatted to exFat so that it can be read under either Windows or Linux.
- Open Oracle VM VirtualBox Manager, and change the location it stores VM’s:
  File → Preferences → General
  Change “Default Machine Folder” to be /media/<user>/<USB Drive name>
- In Oracle VM VirtualBox Manager create a VM.
  - Edit VirtualBox's preferences and download the Extension Pack for Virtual Box from virtualbox.org. Note installed version is 4.3.40.
  - Edit the settings of the VM (when the VM is powered off) under USB select "Enable USB 2.0 (EHCI) Controller"
- On a different PC you want to develop on, in the VM Manager select Machine → Add, and browse to your VM's folder on your USB stick.
- **Note that you must be extra careful never to pull the USB stick when the VM is running because this could corrupt the state of your VM.**
- You may also find it faster to shut-down and fully restart the VM than having the VM suspend to disk. Plus, this is one less thing to go wrong with writing to the USB drive.
- When switching a VM between computers, some settings may need to be updated such as the network adapter used for bridging, shared folders, and so forth.

2. Basic Wiring and First Boot

1. Connect the BeagleBone Green to the host PC is using a single micro-USB cable.
    - Plug the USB-A end into the PC and the USB-micro end into the port on the BeagleBone beside the Ethernet jack, underneath the board.
    - Note that the Zen cape (top board) also has a USB-micro connector on the top; this is explained in the next section.
    - If you are using a BeagleBone *Black*, then you must use a USB-mini connector.
2. You'll see the lights on the BeagleBone start flashing!
3. Other connections, such as the Zen cape's USB-micro, or an Ethernet cable are discussed in later sections and guides.

Note that the USB-micro connection to the BeagleBone provides the following:

- Power to the BeagleBone
    - If you have extra hardware plugged into the BeagleBone, you may find that it begins pulling too much current from the PC's USB port. In which case you may need a powered hub, or to run the system off a USB power adapter (plug it into the wall and it gives power to a USB-A port).
- Ethernet access between the host and target over USB.
- Mapping the BeagleBone as a mass storage device to the host PC.

We only are using the first two of these during this course.

3. Screen

Screen is a tool which allows you to interact with a serial port through Linux.

1. Zen Cape's TTL to USB setup:
   - Connect a second micro USB cable to the Zen Cape (top micro USB slot; same side of board as the Ethernet connection). This is the TTL serial to USB adapter.
   - If using a virtual machine, you may need to map the adapter to the VM. In VMware Player:
     - Find the icon in upper right (on the tool bar) which reads “future devices ft230x basic uart”.
     - Right click it, and select the connect option.
   - In Linux, just after connecting the device you can see if it has been configured correctly by typing:
     ```
     $ dmesg
     ```
     - This shows messages produced by the kernel. You should see the last few lines being:

```
[248996.913965] usb 2-2.1: new full-speed USB device number 4 using uhci_hcd
[248997.142367] usb 2-2.1: New USB device found, idVendor=0403, idProduct=6015
[248997.142370] usb 2-2.1: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[248997.142371] usb 2-2.1: Product: FT230X Basic UART
[248997.142372] usb 2-2.1: Manufacturer: FTDI
[248997.142373] usb 2-2.1: SerialNumber: DA01QHN9
[248998.220827] usbcore: registered new interface driver usbserial
[248998.220854] usbcore: registered new interface driver usbserial_generic
[248998.220886] usbserial: USB Serial support registered for generic
[248998.226983] usbcore: registered new interface driver ftdi_sio
[248998.227000] usbserial: USB Serial support registered for FTDI USB Serial Device
[248998.227037] ftdi_sio 2-2.1:1.0: FTDI USB Serial Device converter detected
[248998.227124] usb 2-2.1: Detected FT-X
[248998.229938] usb 2-2.1: FTDI USB Serial Device converter now attached to ttyUSB0
```

     - This shows that the device is connected to `/dev/ttyUSB0`. If you have multiple devices connected, yours may connect to `ttyUSB1` or higher.

2. Check `/dev/ttyUSB0` exists with the following command (expected output shown on next line):
   ```
   $ ls -al /dev/ttyUSB0
   crw-rw---- 1 root dialout 188, 0 Sep  1 09:57 /dev/ttyUSB0
   ```

3. Install the `Screen` program to be able to read/write to the serial port:
   ```
   $ sudo apt-get install screen
   ```

4. Start `Screen` with the following command:
   ```
   $ sudo screen /dev/ttyUSB0 115200
   ```
   - Where `/dev/ttyUSB0` is the port. May be `ttyUSB1`, etc.
   - `115200` is the “baud rate”, or the speed of the connection. 115200 is the default serial port speed for the BeagleBone.
   - NOTE: If in SFU labs, you do not have root permission to host OS. Do not try any `sudo` commands in the *host OS* in the lab! Our IT staff will be automatically notified and they may contact you about it. Use `sudo` only on your PC or in the VM.

- If you want to use Screen inside the host OS in the lab, you may have access to `/dev/ttyUSB0` already so just type the command without `sudo`.

5. Press Enter to see text on the serial port if you are connecting to a target device which is already running.
   - User name is `root` and there is no password (just press enter).
   - Note that the `root` user's home directory is `/root/`, whereas most other users (such as one named `brian`) would be in the folder `/home/brian/`. This is the normal way Linux works with home directories.
6. Screen commands:
   - Help: Ctrl-A then ?
   - **Quit: Ctrl-A then **
7. Trouble shooting
   - After launching Screen, if you immediately see the message:
     `[screen is terminating]`
     it may mean that you do not have access to `/dev/ttyUSB0`. Either change the permissions on `/dev/ttyUSB0` to be read/write by all users, or launch `screen` with "sudo". Note that in the SFU labs, you do not have sudo (root) access.
     - Check permissions with
       `$ ls -la /dev/ttyUSB0`
   - If you briefly see the following message at the bottom of the screen:
     `Cannot exec 'ttyUSB0': No such file or directory`
     it likely means that you have not correctly connected the Zen Cape's USB-to-TTL-serial-adapter, or if you are using a virtual machine, that you have not mapped it to the virtual machine correctly. For the virtual machine, try disconnecting the USB to TTL adapter from the VM, and then reconnecting it.
   - If your Screen session locks up or begins dropping characters or corrupting text, try:
     - In a VM, disconnect the USB to TTL adapter from the VM and/or physically disconnect the USB cable. Perhaps try plugging it back into another USB slot.
     - Ensure `Screen` is running with the correct baud rate (115200).
     - Ensure your VM has more than the minimum video memory (if there's a setting for this).
     - Try increasing its access to 3D acceleration and the number of CPU cores (if possible).
     - Try closing the `screen` session (Ctrl-A then \), and restarting the screen session. Sometimes under VirtualBox it seems that closing `screen` also locks up, but it then resolves within a minute and successfully quits `screen`.
   - If trying to reconnect gives the error: "Failed to attach the USB device FTDI TTL232R-3V3..." and "... is busy with a previous request", with VirtualBox then you may need to kill the VBoxSVC.exe process in the Host OS, and/or reboot the host OS.
   - Your USB adapter may also come up on /dev/ttyACM0. You should still use /dev/ttyUSB0. If it only comes up as /dev/ttyACM0, then it should be OK to use it.

4. Connecting via SSH and NFS

4.1 Networking

Follow the steps in the Networking guide posted on the course website.

4.2 SSH

1. SSH from your host PC to the target device:
   ```
   $ ssh root@192.168.7.2
   ```
   - Where the IP address is the address of the target device (BeagleBone). You can find this by running `ifconfig` on the target over the serial connection (using `screen`).
   - The "`root@`" tells SSH we want to connect as a specific user, the user named root, rather than the user name we logged into the host PC with. The root password is empty.
   - You may be asked to accept a security certificate; type "yes".
   - If you have made significant changes to the target device (such as reinstalling its operating system or file system), it may require you to remove a saved security certificate from the host PC before allowing you to SSH to the target. It will give you the command to execute in the error message.
2. Via the SSH connection, you can do anything would do via the serial connection. For example:
   ```
   # echo I am connected via SSH
   # cd /proc
   # cat uptime
   # cat cpuinfo
   ```
3. In general, prefer using the SSH connection over the serial connection because it is usually faster and less error prone.

4.3 NFS

Follow the steps in the NFS guide posted on the course website.

5. Cross-compile Setup

5.1 Setup folders

1. On the host PC, create a directory for your work and for downloading (public):
   ```
   $ cd ~
   $ mkdir -p cmpt433/work
   $ mkdir -p cmpt433/public
   ```
   - Change `~/cmpt433/public` to have global read/write permissions:
     ```
     $ chmod a+rw ~/cmpt433/public
     ```
   - Make sure none of your source code, makefiles, or the like is in the public directory. This directory can be read by everyone, so someone could take your work and hand it in, causing great problems for both students! It is OK, though, to have compiled programs and downloadable images (for U-Boot, the kernel, and the file system) in the public directory.
2. Install the cross-compiler and required tools:
   ```
   $ sudo apt-get install gcc-arm-linux-gnueabihf
   $ sudo apt-get install binutils-arm-linux-gnueabihf
   ```
   - The `hf` in the above commands means hardware floating point.[1]
3. Check tools installed with the following. It should display the compiler's version info:
   ```
   $ arm-linux-gnueabihf-gcc --version
   arm-linux-gnueabihf-gcc (Ubuntu/Linaro 5.4.0-6ubuntu1~16.04.1) 5.4.0 20160609
   Copyright (C) 2015 Free Software Foundation, Inc.
   This is free software; see the source for copying conditions.  There is NO
   warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   ```

1 If you are targeting an old release for the BeagleBone which does not support hardware floating point, you'll want to install: `gcc-arm-linux-gnueabi` and `binutils-arm-linux-gnueabi`

5.2 Build for Host vs Target

1. Create a new directory, called `myApps`, inside your work folder for all the applications you'll be writing:
   ```
   $ cd ~/cmpt433/work
   $ mkdir myApps
   ```
2. Create a `quickStart` folder inside `myApps`.
3. Create a `helloworld.c` program inside the `quickStart` folder. Use the `gedit` editor built into Ubuntu to edit the program. Note that the `&` at the end of a command makes it execute in the background, which here allows you to keep using your terminal while `gedit` is running.
   ```
   $ gedit helloworld.c &
   ```
4. Write the hello world program in C and save the file:
   ```
   #include <stdio.h>
   int main(char* args[])
   {
         printf("Hello embedded world!\n");
         return 0;
   }
   ```
5. Compile and run the program **for the host OS**:
   ```
   $ gcc helloworld.c -o helloworld_host
   $ ./helloworld_host
   ```
6. Compile the program **for the target OS** (Linux on the BeagleBone):
   ```
   $ arm-linux-gnueabihf-gcc helloworld.c -o helloworld_target
   ```
   - Try running the `helloworld_target` application on the host OS. What happens?
7. Analyze the two executables you have just compiled using `readelf`:
   ```
   $ readelf -h helloworld_host
   …
   Machine:  Advanced Micro Devices X86-64
   …
   Flags:    0x0

   $ readelf -h helloworld_target
   …
   Machine:  ARM
   …
   Flags:    0x5000400, Version5 EABI, hard-float ABI
   ```
   - Use `readelf` when you need to identify which OS an executable is compiled for.
8. Troubleshooting
   - When running `arm-linux-gnueabihf-gcc`, if you get the error `sys/cdefs.h` not found, you may be missing the 32-bit libraries. Try:
     ```
     $ sudo apt-get install libc6-dev-armhf-cross
     ```
     Or use the Aptitude package manager:
     ```
     $ sudo apt-get install aptitude
     $ sudo aptitude gcc-arm-linux-gnueabihf binutils-arm-linux-gnueabihf
     ```

5.3 Running via NFS

1. Complete the NFS guide posted on the website to map the `~/cmpt433/public` folder on the target board.
2. On the host PC, copy the `helloworld_target` executable to the `~/cmpt433/public/` folder.
   ```
   $ cd ~/cmpt433/work/myApps/quickStart
   $ cp helloworld_target ~/cmpt433/public/
   ```
3. On the target, change to the mounted NFS folder:
   ```
   # cd /mnt/remote
   ```
4. On the target, run the executable:
   ```
   # ./helloworld_target
   ```
5. Congratulations! You've done some embedded development!
6. Repeat these steps for the `helloworld_host` executable and see the result. Should the host version work on the target?
7. Troubleshooting:
   - If you get a “File not found” error, even though your program is obviously in the correct spot, it may mean that you have a mis-match between the cross-compiler on your host and your target’s OS in terms of hardware vs software floating point. Use `readelf` on your executable to see if it uses software or hardware floating point (the `Flags`). You can check what your target requires by compiling the Hello World example program natively on the target (using `gcc` directly on the BeagleBone), and run `readelf` on the program it generates.