Quick Start Guide for BeagleY-AI

by Brian Fraser

Last update: Jan 1, 2025

Guide has been tested on

BeagleY-AI (Target): **Debian 12.8**
PC OS (host): **Debian 12.8** (or higher)

This document guides the user through

1. Installing Debian on the computer 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

Formatting

1. Commands for the host Linux's console are show as:
```
(host)$ echo "Hello PC world!"
```
2. Commands for the target (BeagleY-AI) Linux's console are shown as:
```
(byai)$ echo "Hello embedded world!"
```
3. Almost all commands are case sensitive.

Revision History

- Jan 2025: Target the Beagle-Y AI

1. Host OS Setup

1. **Linux Debian 12.8+ ("Bookworm")** 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 **Debian 12.8** from its official site: https://www.debian.org/
     Click the "Download" button on the right. Ensure you get the correct version for your processor; likely **amd64** (PC) or **arm64** (mac)
   - Use of Ubuntu is *not* recommended because of possible version conflicts of the C libraries between the target vs host. Since the target runs Debian 12.8, that is a solid choice for your host.
2. If you currently run Windows or macOS, you may install Linux in a virtual machine (VM). Running inside a VM means one extra level of configuration is required, but it is an effective setup for this course.
   - Under macOS (especially for Apple processors), you may want to run VMware Fusion. It's free, but requires you to make a login. Students have also had success with UTM (free on website; paid in Mac App Store).
   - Under Windows VMware Worstation Pro or VirtualBox work (both free; may require login). This YouTube playlist features some videos showing how to install a VM. You can also use the Windows Subsystem for Linux (WSL) which is a little more coupled to Windows and slightly more awkward to work with in some ways (USB), but which does not require a VM. If interested, see WSL guide on course website.

     I recommend VMware because under Windows because it runs well with Hype-X if you have previously installed WSL 2 or Docker.
     - If you use VMware, 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.
     - If given the choice, select "Graphical Install" (over just "install") so you get a GUI.
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.
   - 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 can improve performance.
4. Once you have installed Debian, you will want your user to be able to use the **sudo** command. Try the following command:

   ```
   (host)$ sudo ls
   ```

   If it failed with message about "... is not in the sudoer file", then try the following ([1]):

   ```
   (host)$ su
   (host)# EDITOR=nano visudo
   (host)# /sbin/visudo
   ```

   (Unnecessary on Debian)

   Add the following to the end of the file, changing "`brian`" to your current user name:

[1] Linuxize, "How to add user to sudoers in Ubuntu": https://linuxize.com/post/how-to-add-user-to-sudoers-in-ubuntu/

```
brian   ALL=(ALL) NOPASSWD:ALL
```

To save and exit press CTRL+x, then type “Y” and enter to save the file (“buffer”).

Exit the `su` prompt (`` `exit` ``) and retry the `` `sudo ls` `` command!

5. If you are running a 4K display, you may want to change the UI scaling inside your VM for the Gnome graphical environment:

```
(host)$ gsettings set org.gnome.settings-daemon.plugins.xsettings overrides \
  "[{'Gdk/WindowScalingFactor', <2>}]"
(host)$ gsettings set org.gnome.desktop.interface scaling-factor 2
```

6. If running in a virtual machine, I recommend turning off power saving features in Linux as I have found this can lock-up the VM when it tries to enter power-saving mode:
   - In your Linux VM, go to Settings > Power
   - Black Screen: Never
   - Automatic Suspend: Off
7. Install basic software
   - **Update**

     ```
     (host)$ sudo apt update
     (host)$ sudo apt upgrade
     ```

   - **Install VS Code**[2]
     Download the .dep package from https://code.visualstudio.com/download .
     If the installer asks, it’s OK to add the VS Code update site to the local configuration.

     ```
     (host)$ sudo apt install ~/Downloads/code_*
     (host)$ code
     ```

   - **Git**

     ```
     (host)$ sudo apt install git
     ```

   - **SFU VPN**
     On my computer, if I have my host OS (Windows) running the SFU VPN then I am able to connect to our GitLab server. SFU allows CS students to use the VPN.

     If you want native Linux support you may try:

     ```
     (host)$ sudo apt install openfortivpn
     ```

     Each time you need to connect to an SFU system requiring the VPN, you can run with:

     ```
     (host)$ sudo openfortivpn vpn.its.sfu.ca:10433 -u <YourSfuUserId>
     ```

     When prompted, enter your SFU password, and then a MFA code.
     Suggestion: Make a script in your home folder for this.

8. Troubleshooting:
   - If having problems with `apt` 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:

     ```
     (host)$ sudo software-properties-gtk
     ```

   - If your computer crashes when launching your VM, try disabling any automatic USB device

[2] It’s possible to install VS Code via Snap; however, this is suspected of making some VMs unstable and freezing. Commands to install:

```
(host)$ sudo apt install snapd
(host)$ sudo snap install --classic code
```

redirects to your VM.

- If you are having troubles getting Linux to run in a VM under windows with Hyper-X/Docker/…, try:
  - Go to Start, then search for “Turn Windows features on or off”
  - Enable all of the following (will likely make VirtualBox not work; for VirtualBox to work you may need to disable Windows Hypervisor Platform, and/or Hyper-X):
    - Hyper-V
    - Virtual Machine Platform
    - Windows Hypervisor Platform
- If you install your host OS and it has no graphical user interface, then you may have needed to select “Graphical Install” during the installation instead of just “Install.”
- When installing VS Code, you may get the error “`Download is performed unsandboxed as root as file '/home/brian/Downloads/code_1.96.3-1736454372_amd64.deb' couldn't be accessed by user '_apt'. - pkgAcquire::Run (13: Permission denied)`”. However, this seems not to prevent it successfully installing.

2. Assemble the BYAI boards

You will need to screw together the BYAI with the Zen Hat (with LCD) and the mounting plate. Follow directions on the course website for this.

3. Create micro SD Card and Boot

There is no on-board storage on the BeagleY-AI; it needs a micro SD (uSD) card to boot.

1. On you host, under Linux, install the `bb-imager` application:
   - Download the correct `.deb` installer: https://beagley-ai.beagleboard.io/bb-imager/
     - On Linux:
       If you are on an ARM64 system, download: `bb-imager_2.0.0_arm64.deb`
       If you are on an x86 system, download: `bb-imager_2.0.0_amd64.deb`
   - Install and run:
     ```
     (host)$ sudo apt install ~/Downloads/bb-imager_2.0.0_a??64.deb
     (host)$ bb-imager
     ```
   - If using Windows directly to write the uSD card, I found `bb-imager` did not work for me (not trusted by Windows, detected as a virus by my virus scanner). Instead I installed it in my VM and then it worked fine. Another option is the Balena Etcher program and then manually downloaded the necessary BYAI image, and edit the `sysconfig.txt` file.
2. Follow these directions from beagleboard.org to create a customized uSD card.
   - Suggested version: “BeagleY-AI Debian v6.1.x Minimal”
     - It’s fine to install either the XFCE or Minimal versions. The XFCE supports a graphical interface if using an HDMI display. We can get away with the smaller Minimal image.
   - Make the following OS customizations:
     - **Under General**
       - Set the username to be your name.
       - Set the password to be something unique; do not make it the same as your SFU password.
       - If desired, set the wireless LAN settings.
     - **Under Services**
       - Enable SSH
       - Set to “Use password authentication”
3. Once it has created the uSD card and verified, remove it from the computer and insert it into the powered-off BeagleY-AI.
4. Connect the BeagleY AI (BYAI) to the host PC is using a USB-C cable.

   a) Plug the USB-A end of the cable into the PC (or use a USB-C to C cable, not provided, if needed)
   b) Plug the USB-C end into the port on the BeagleY-AI.

   You’ll see the lights on the BeagleY-AI start flashing!
   Other connections, such as the Ethernet cable, are discussed in later sections and guides.
5. When powering on the BeagleY-AI, look at the LED on the bottom board (the BeagleY-AI):
   - LED is red (solid or flashing) during initial startup (~30s).
   - LED flashes green during Linux running.

- Linux will apply some OS customizations and then reboot.
- LED is red (flashing or solid) during second startup (~30s).
- LED flashes green during second Linux startup and run.
- LED continues to flash a green heartbeat.

6. Note that the USB-C connection to the BYAI provides the following:
   - Power to the BYAI

     If you have extra hardware plugged into the BYAI, 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 port).
   - Network access between the host and target over USB via “Ethernet over USB”.
   - Mapping (part of) the BeagleY AI as a mass storage device to the host PC.

   We only are using the first two of these during this course.
7. Troubleshooting
   - When running `bb-imager`, under Storage it shows “No storage devices found” then try:
     - If in a VM ensure that you have the uSD card reader mapped to the VM.
     - Try disconnecting the uSD card reader and reconnecting it to the computer using another USB port (if applicable).
     - Ensure your uSD card is inserted into the reader.
   - If the board’s LED stays on solid-red, it likely means that there is either no uSD card inserted, or the uSD card does not have a valid bootable image. Try reflashing your uSD card.
   - If the board’s LED cycles between flashing red and green more than twice, it may mean the board is unable to startup correctly.

     Ensure that you have enough power to the BeagleY-AI. It takes about 1.0amps (1000 mA) to run the board which may be more than some USB ports can handle. Try another USB port, a powered USB hub, another computer, or a good USB power adapter.
   - If the board is unable to boot after reflashing the uSD card and using a good USB power adapter, it may be useful to see the boot sequence using a serial debug device.
     - You’ll know there is a problem because it keeps flashing the LED as red (perhaps sometimes being solid).
     - Check that the board has enough power by plugging it into a good USB power device.
     - See instructor or the Serial Guide for more.
   - While installing bb_imager, you may see the error: “`N: Download is performed unsandboxed as root as file '/home/.../Downloads/bb-imager_2.0.0_amd64.deb' couldn't be accessed by user '_apt'. - pkgAcquire::Run (13: Permission denied)`”. In spite of this error, the tool seems to run fine.
   - If you encounter installation troubles for bb-image, you may want to try installing some of the bb-imager build dependencies found here: https://github.com/beagleboard/bb-imager

4. Connecting via SSH and NFS

4.1 Networking

Right *now*, follow **Sections 1 and 2** in the **Networking** guide posted on the course website, then come back here.

4.2 SSH

1. SSH from your host PC to the target device:
   ```
   (host)$ ssh username@192.168.7.2
   ```
   - Replace `username` with the user name you entered when configuring your uSD card.
   - If needed, replace the IP address of the target device.
   - The "`username@`" tells SSH we want to connect as a specific user, the user named `username`, rather than the user name we logged into the host PC with.
   - You may be asked to accept a security certificate; type "yes".
   - If you reflash the micro SD card, or plug in a different target device to your computer, SSH will 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. Optional: Setup an OpenSSL authentication key on the target so you don't need to type a password each time you SSH:

   -
     ```
     (host)$ ssh-keygen -t ed25519
     ```
     Do not replace any existing .ssh/id_ed25519 files.
     OK to just press enter to prompts.
   - Copy the
     ```
     (host)$ ssh-copy-id username@192.168.7.2
     ```
   - SSH to the target to test it no longer asks for your password:
     ```
     (host)$ ssh username@192.168.7.2
     ```
3. Optional: Use `sshpass` to SSH to the target without having to retype password. Note that this then stores the password in clear text on your computer, so don't use this if you use this password *anywhere* else!

   ```
   (host)$ sudo apt install sshpass
   (host)$ sshpass -p temppwd ssh debian@192.168.7.2
   ```

   If this command fails, try first connecting via `ssh` command.
4. Via the SSH connection, you can do anything would do via the serial connection. For example:
   ```
   (byai)$ echo I am connected via SSH
   (byai)$ cat /proc/uptime
   (byai)$ cat /proc/cpuinfo
   ```
5. Check the version of software image installed on your BeagleY-AI with the following:
   - Version of Debian:
     ```
     (byai)$ cat /boot/firmware/ID.txt
     ```

```
BeagleBoard.org Debian Bookworm Minimal Image 2024-09-06
```

- Version of Linux Kernel:
  ```
  (byai)$ uname -a
  Linux BeagleBone 6.1.83-ti-arm64-r63 #1bookworm SMP PREEMPT_DYNAMIC Wed
  Jul 10 23:00:56 UTC 2024 aarch64 GNU/Linux
  ```

6. **Troubleshooting**

- `sshpass` will fail (doing nothing, showing no output) if you have not yet SSH'd into the target and it needs you to accept the identification information for the device.
  If `sshpass` fails, first connect with `ssh` and then retry using `sshpass`.
- If your BYAI shows an older version, then you may want to reflash your microSD card.

4.3 NFS

Right *now*, follow the steps in the NFS guide posted on the course website.

Hint: You'll now have done a lot of setup steps on the BYAI. You may want to create a setup script which can re-do these operations quickly if needed (if you have to reflash your uSD card). For example, here is one student's setup script targeting an older version of the target.

5. Cross-compile Setup

5.1 Setup folders & Tools

1. On the *host* PC, create a directory for your work and for downloading (public):
   ```
   (host)$ mkdir -p ~/cmpt433/work
   (host)$ mkdir -p ~/cmpt433/public
   ```
   - Change `~/cmpt433/public` to have global read/write permissions:
     ```
     (host)$ 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.
   - **Strong Suggestion!**
     Create a GitHub repo and check it out as `cmpt433/work/` instead of an empty folder:
     ```
     (host)$ git clone <github URL> ~/cmpt433/work
     ```
2. Install the cross-compiler and required tools:[3]
   ```
   (host)$ sudo apt install gcc make cmake
   (host)$ sudo apt install gcc-aarch64-linux-gnu binutils-aarch64-linux-gnu
   ```
   - If you choose to install a non-standard version of the compiler, create a link in the file system to redirect to your newly installed cross compiler:
     ```
     (host)$ cd /usr/bin
     (host)$ sudo ln -s aarch64-linux-gnu-gcc-10 aarch64-linux-gnu-gcc
     ```
3. Check tools installed with the following. It should display the compiler's version info:
   ```
   (host)$ aarch64-linux-gnu-gcc --version

   aarch64-linux-gnu-gcc (Debian 12.2.0-14) 12.2.0
   Copyright (C) 2022 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.
   ```

[3] You can choose to install a different version of the GCC cross compiler; however, going with a very new version may create binaries which depend on a newer version of GLibC than is installed on the target. To install the latest GCC, use:
```
(host)$ sudo apt install gcc-aarch64-linux-gnu
```

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:
   ```
   (host)$ mkdir -p ~/cmpt433/work/myApps/quickStart
   (host)$ cd ~/cmpt433/work/myApps/quickStart
   ```
2. Create a `helloworld.c` program inside the `quickStart` folder.
   - Use VS Code (`code`) to edit the program:
     ```
     (host)$ code helloworld.c
     ```
   - If asked, install any recommended plugins.
3. Write the hello world program in C and save the file:
   ```
   #include <stdio.h>
   int main(int argc, char* args[])
   {
         printf("Hello embedded world!\n");
         return 0;
   }
   ```
4. Compile and run the program **for the host OS**:
   ```
   (host)$ gcc helloworld.c -o helloworld_host
   (host)$ ./helloworld_host
   ```
5. Compile the program **for the target OS** (Linux on the BeagleY-AI):
   ```
   (host)$ aarch64-linux-gnu-gcc helloworld.c -o helloworld_target
   ```
   - Try running the `helloworld_target` application on the host OS. What happens?
6. Analyze the two executables you have just compiled using `readelf`:
   ```
   (host)$ readelf -h ./helloworld_host
   …
   Machine:  Advanced Micro Devices X86-64

   (host)$ readelf -h helloworld_target
   …
   Machine:  AArch64
   ```
   - Use `readelf` when you need to identify which OS an executable is compiled for.

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.

   ```
   (host)$ cd ~/cmpt433/work/myApps/quickStart
   (host)$ cp helloworld_target ~/cmpt433/public/
   ```

3. On the target, change to the mounted NFS folder:

   ```
   (byai)$ cd /mnt/remote
   ```

4. On the target, run the executable:

   ```
   (byai)$ ./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 mismatch 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 BeagleY-AI), and run `readelf` on the program it generates.

6. Done

If you are using a VM, now that you have it configured, you may want to take a snapshot of it in case something goes wrong in the future.

- If running VMware:
  - Shutting down your VM (power off) then tell your Linux VM to shutdown. Make sure you tell Linux to shutdown, rather than clicking the X on the VM to close the window.
    (Note that if you are just suspending the VM, clicking the X and choosing suspend is a good choice for day-to-day use).
  - Open VMware Workstation, click on your VM
  - On the menu select VM > Snapshot > Take snapshot
- If running VirtualBox, it also supports taking snapshots within its interface.
  - Note that a snapshot is not a full copy of the VM, but a diff that can be used to restore the VM to this state.