Beagle Entertainment System v0.8 User Guide

Andrew Henderson
www.beaglesnes.org

WWW.BEAGLESNES.ORG

BEAGLE ENTERTAINMENT SYSTEM

Contents

BEAGLE ENTERTAINMENT SYSTEM

1 — The Beagle Entertainment System

1.1 Overview

Beagle Entertainment System (commonly abbreviated as "*BES*")is a complete software system that turns the ARM-based BeagleBone Black (BBB) hardware platform[1] into a dedicated device capable of executing software originally developed for the Nintendo Entertainment System®(NES), Super Nintendo Entertainment System®(SNES), Gameboy®(both the original and "color" versions) and the Gameboy Advance®(GBA) video game consoles[2]. BES allows you to play software titles for all of these consoles on your television or computer monitor via the HDMI output of your BBB.

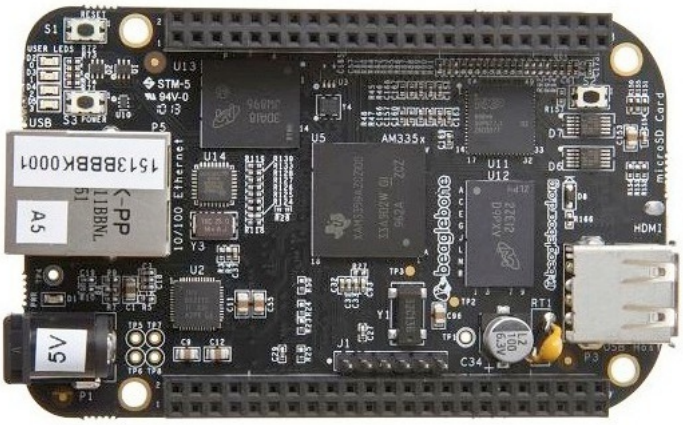

Figure 1.1: The BeagleBone Black hardware platform.

BES originally began as a single emulator for the SNES: BeagleSNES. BeagleSNES was a course project for a graduate embedded systems design class. While the scope of the project was to port the SNES9X emulator to the BeagleBoard-xM and then tune the board's Linux kernel for

[1] Learn more about the BeagleBoard and BeagleBone ARM-based single board computer families at `www.beagleboard.org`.

[2] Nintendo®, Nintendo Entertainment System®, Super Nintendo Entertainment System®, Gameboy®, and Gameboy Advance®are all registered trademarks of Nintendo Co. Ltd. and its subsidiary companies.

performance, things got a little out of hand. The project kept growing in complexity and scope. Over the span of four months, it matured into much of BES that you see today. The bootloader, Linux kernel, file system, and emulator of BeagleSNES were all customized for functionality, stability, and performance. These components were further customized when the project was later ported to the BBB. Later, even more features were added as additional emulators and configuration options were added to the project, leading up to BES.

The source code for BES, as well as the latest instructions for building, installing, and using the software, are freely available from `www.beaglesnes.org`. Almost all of BES is licensed under the GNU Public License (version 3) as open source software (OSS)[3]. What does this mean? It means that you are welcome to examine the code, modify it, learn from it, and even use it in projects of your own. But, you may not sell the system if it contains the emulator.

The hardware of these four consoles (CPU, GPU, memory, audio chipset, etc.) are quite different from the hardware of BBB. Only the GBA uses a 32-bit ARM instruction set like the BBB does. Because the software for these consoles will never execute natively on the BBB, BES must *emulate* the hardware of the consoles in software. The inputs, outputs, and internal state of each piece of console hardware is represented as high-level source code (C and C++ code, in the case of BES). For example, the 16-bit microprocessor of the SNES (the WDC 65C816) runs at a clock speed of 3.58 MHz. This is much slower than the 32-bit ARM CPU of the BBB (AM335x 1 GHz Cortex-A8), meaning that there are enough resources available to dynamically translate every SNES program instruction, at the time that it would be executed on the SNES, into one or more ARM instructions and then execute them. This *just-in-time dynamic translation* approach has considerable overhead, but it is very effective in situations where the emulated system is much slower than the system performing the emulation.

Figure 1.2: The Super Nintendo Entertainment System®video game console.

Many Linux-based emulators currently exist, and BES leverages this existing work rather than “reinventing the wheel". For SNES titles, BES uses the SNES9X emulator[4]. For NES, the Nestopia emulator[5] is used. For Gameboy, Gameboy Color, and GBA, the VBA-M emulator[6] is

[3]BeagleSNES is based on the SNES9X emulator, which is licensed under a custom, non-commercial license. BeagleSNES is a subset of the BES source code. Because of this, BES is largely a self-contained binary that launches an external BeagleSNES binary when playing SNES titles.

[4]http://www.snes9x.com/

[5]http://nestopia.sourceforge.net/

[6]https://sourceforge.net/projects/vbam/

used. These were chosen for the following reasons:

- All source code for these emulators is available, and all optimizations and additions made to them as part of BES can be freely released as long as the terms of the GPL and SNES9X licenses are followed.
- The source code for these emulators is all cross-platform, meaning that their implementations are not tied to a particular display library or CPU architecture.
- Their emulation compatibility is quite good, allowing them to run many different pieces of console software accurately.

While many architecture-specific performance optimizations exist in these emulators (such as using x86-based assembly cores to accelerate the emulation), many of these improvements can't be used with the ARM-based BBB. This limits overall emulation speed, but the performance is generally acceptable. This approach isn't perfect, as the overhead can be prohibitively expensive computationally. Depending upon the console, video frames may be dropped to reach the proper execution speed. In general, Gameboy and Gameboy Color titles will always run at full speed without dropping frames, NES and SNES titles will drop a few frames here and there, and GBA titles will drop quite a few frames (but still remain quite usable).

1.2 Features of BES

You could always apt-get a few emulators and use them to run old software on your BBB, so what makes BES so special? A few of the features that BES provides are:

- **Higher performance emulation.** Each emulator is optimized for the BBB platform, both in the kernel and in userspace. Custom patches have been applied to the kernel, the emulators have been profiled and optimized, and the various tweaks and tuning for each emulator have already been performed.
- **Uniform pause and save state functionality.** Each emulator is modified to provide a pause pop-up menu that allows you to exit the emulator or save/restore your game state.
- **A front-end GUI for game selection.** Quickly and easily select your favorite games, and view game information and screenshots of the last saved state for each game.
- **Web-based system configuration.** Use the web interface to map buttons on your controllers, add/remove ROMs, and edit the game information shown in the front-end GUI. You do not have to hand-edit configuration files on your BBB, or even touch the BBB's filesystem!
- **OpenGL ES hardware scaling.** Each emulator has been modified to render into an OpenGL ES texture, which is them mapped to the entire size of the screen. Whether you are running on an LCD cape or your HDMI television, scaling and filtering is done in hardware to give you excellent video quality at all resolutions with minimal overhead penalty to performance.
- **A variety of controller options.** Want to use a USB gamepad to play? GPIO-based buttons? Native SNES gamepads? There is support for all of these options, allowing you to use BES as a base for your own handheld or console projects. Even better, all emulators will work with these options automatically, meaning that you configure your controller options once and everything "just works" across all of the emulators.

1.3 Using This Manual

This guide was written to help you to install, configure, and use BES. For any important aspects of the system that warrant additional attention, this manual uses the following box to highlight them:

Figure 1.3: Advanced users interested in experimenting with BES for their own projects can use BES with GPIO inputs (upper left), alternative displays (upper left and upper right), native SNES gamepad interfacing (bottom left), or even create a custom retrogaming console (bottom right).

Be sure to pay attention to these boxes as you read through the manual. They will alert you to any important or useful information that you should be aware of.

For any commands executed on the command line, this manual uses the following box:

```
username@host$ sudo ./execute_this_command
```

Each section of the guide contains information on a different aspect of BES. Section 1 is the introduction that you are reading right now. Section 2 describes how to download and install the pre-made file system image of BES. For most end-users, this section will be the place to start. Section 3 provides instructions on how to add ROMs to the BES system and configure the game selection GUI and gamepad button mapping. Section 4 is a troubleshooting guide to help you work through some of the common issues that people come across.

Enjoy using BES!

1.4 BeagleSNES License

The bootloader, Linux kernel, and GNU utilities used within BES are all licensed under the GPL. These components can be freely modified, shared, and even sold, as long as the terms of the GPL are followed. Feel free to use them as a base for your own projects!

The license information for the BeagleSNES emulator application can be seen in figure 1.3. This license is a slightly modified version of the SNES9X license, which directs you to the stock SNES9X license for all of the details. It allows you to look at, learn from, and use BeagleSNES

for personal use, but not to exploit it for commercial gain. This is NOT GPL licensed software, but it is freeware. All other emulators within BES are licensed under the GPL.

```
1  BeagleSNES - Super Nintendo Entertainment System (TM) emulator for the
2  BeagleBoard-xM and BeagleBone Black platforms.
3
4  (c) Copyright 2013 - 2014    Andrew Henderson (hendersa@icculus.org)
5
6  BeagleSNES is a GUI front-end that is integrated into to the Snes9x
7  emulator, along with some hacks that shut off unneeded functionality
8  and tune the performance for execution on the BeagleBoard-xM and
9  BeagleBone Black platforms.
10
11 Please refer to the snes9x-license.txt file for more information on
12 the authors and license pertainting to the Snes9x codebase.  Specific
13 ports contains the works of other authors. See headers in individual
14 files.
15
16 BeagleSNES homepage: http://www.beaglesnes.org/
17 Snes9x homepage: http://www.snes9x.com/
18
19 Permission to use, copy, modify and/or distribute BeagleSNES in both
20 binary and source form, for non-commercial purposes, is hereby granted
21 without fee, providing that this license information and copyright
22 notice appear with all copies and any derived work.
23
24 This software is provided 'as-is', without any express or implied
25 warranty. In no event shall the authors be held liable for any damages
26 arising from the use of this software or it's derivatives.
27
28 BeagleSNES is freeware for PERSONAL USE only. Commercial users should
29 seek permission of the copyright holders first. Commercial use includes,
30 but is not limited to, charging money for BeagleSNES or software derived
31 from BeagleSNES, including BeagleSNES or derivatives in commercial game
32 bundles, and/or using BeagleSNES as a promotion for your commercial
33 product.
34
35 The copyright holders request that bug fixes and improvements to the code
36 should be forwarded to them so everyone can benefit from the modifications
37 in future versions.
38
39 Super NES and Super Nintendo Entertainment System are trademarks of
40 Nintendo Co., Limited and its subsidiary companies.
```

Figure 1.4: The `beaglesnes-license.txt` file included in the BeagleSNES portion of the BES source code.

BEAGLE ENTERTAINMENT SYSTEM

2 — Installing BES

2.1 Overview

BES is more than just a single application. It is a complete POSIX operating system environment, complete with a custom Linux kernel, a full file system, and configuration files that tell BES how to behave for the end user. Because of the general complexity of such a large system, it can sometimes be difficult to get everything up and running smoothly. This section will tell you how to download and install BES so that you can get started using it as quickly and painlessly as possible.

All source code for the project, as well as a full file system image of a complete BES system, have been made available for download. Most users will only wish to try out BES, rather than hack on its source code, so this section will focus on using the full file system image. Prior to starting, you'll need the following items:

- A BeagleBone Black system. No particular version of the board is required.
- An external power supply to provide full power to the system. Powering via USB will not provide the necessary amount of current.
- A microHDMI-to-HDMI cable (BBB) and a television or monitor capable of displaying the digital video signal. If you don't use a monitor with audio support, BES will let you know that there is no audio present and disable audio support in the software.
- A microSD card with a capacity of at least 8 GB. BES will completely wipe the card, so make sure that it does not contain any data that you want to keep.
- Copies of the software ROMs that you wish to use with BES.
- One or two Tomee™USB SNES Gamepad controller(s)[1]. The BBB only supports using one controller unless you use an external USB hub to provide additional USB ports. *USB controllers other than the Tomee™will work with BES*, but they will most likely need their buttons remapped via BES's web interface.

[1] Several online vendors offer this particular USB controller for sale. It is the only controller that will be officially supported for BES.

While many software ROMs are widely available on the web, possession of these ROMs is generally considered to be piracy of commercial software. No ROMs are provided in the BES file system images, and no help in obtaining ROMs will be given. You're on your own.

2.2 Downloading

BES is available for download as a microSD card image that contains the complete BES system (bootloader, kernel, file system, etc.). This image not only contains a working BES implementation, but also a complete development toolchain and the source code that BES was built from. Direct download links to the latest versions of all BES software components are available at `www.beaglesnes.org`. SourceForge hosts all of the releases for BES[2], but `www.beaglesnes.org` should first be consulted for any additional information that you might need about the latest version of BES prior to downloading. Github hosts the source code for both BES[3] and BES's web configuration tool[4]. In addition, the kernel device tree source code used for BES is also available[5]. The device tree source can be modified to multiplex the pins of the P8/P9 headers of the BBB for whatever configuration suits your own BES-based projects.

2.3 Installing

The easiest way to install BES is to write the full system image to a microSD card and then boot your BBB from that card. The speed class of the card is not all that important, but all of the BES development was performed using a class 10 card. Slower speed classes may increase the time needed for a system boot, but otherwise the impact on the performance of BES is assumed to be negligible.

Some advanced BeagleBone users may already have a working file system, kernel, etc. on their system and only wish to install the BES application. For those people, the application image can be copied into their existing file system. Libraries used by BES (SDL, OpenGL ES, etc.) are not shipped with the application image because you might not have built your particular kernel with the same features that are in the BES kernel (OSS audio, framebuffer console, etc.). It will be up to you to install the needed libraries on your system in a fashion that is compatible with your current kernel features. Beware that you will need to install OpenGL ES libraries for BES, which can be quite difficult to do if your distro does not come with them installed by default. In addition, BES needs to be started at boot. There is a script named *service.sh* that should be called by a boot `cron` job, the `rc.local` script, or other similar mechanism upon system start-up. Modify the path to the BES directory in *service.sh* to point to the path where you have installed BES on your system.

If you are just downloading the base image and writing it to a microSD card, you don't need to worry about any of these setup details. Just write the system image to the microSD card, put it into your BBB system, and you are ready to go!

To copy the full system image to your microSD card, first download the image and then use `unxz` to decompress it. Execute a Unix `dd` command similar to the following to write the image

[2] `https://sourceforge.net/projects/beaglesnes/files/`
[3] `https://github.com/hendersa/bes`
[4] `https://github.com/hendersa/bes-config-node`
[5] `https://github.com/hendersa/bes-dtb-rebuilder`

to your microSD card[6]:

```
username@host$ sudo dd if=bes.v0.8.img of=/dev/sdX bs=1M
```

This command will take a little while to run, so be patient. The output file parameter (`of`) will may vary depending upon what device represents the microSD card on your particular Linux system. For most systems, `/dev/sdX` (`sda`, `sdb`, etc.) will represent the microSD card. On others, you might see `/dev/mmcblkX` instead, where the "X" is a number (0 for the first microSD card, 1 for the second, etc.). It is up to you to determine which it is. Don't guess and try different parameters without knowing for sure! A bad guess might end up with your accidentally overwriting data on some other device in your system! The `lsblk` command can aid you in determining which device belongs to your microSD card.

Once the full system image is copied onto the microSD card, the card will contain one `EXT4` partition, so Windows-based systems will see the card as unformatted. Linux and OS X systems will be able to mount this partition in case you wish to examine its contents. This is not necessary, however, as BES configuration is completed via its web interface. Insert the microSD card with the BES image into your BBB, connect your BBB to your HDMI display, and then power the board up. After about 15-20 seconds, the game selection menu will appear, begin playing background music, and display a welcome message.

When using BES on a BBB, you do *not* have to hold down the "user boot" button during boot. The BES `uEnv.txt` file is written such that the bootloader on the BBB's eMMC understands that it should boot from the microSD card and it will do so automatically.

[6]Windows users can use the Win32 Disk Imager utility to write the image to the microSD card. Download it at: `http://sourceforge.net/projects/win32diskimager`

BEAGLE ENTERTAINMENT SYSTEM

3 — Configuring and Using BES

3.1 Overview

Configuring multiple emulators for optimal performance is a difficult task. Different emulators offer different features, use different default key mappings, and handle games saved on the cartridge (battery-backed SRAM) and save states differently. Add in platform-specific configuration settings for scaling, frame skipping, and audio and it you can easily spend hours trying to create the perfect console experience.

Luckily, BES handles almost all of these details for you. The BBB has a static hardware configuration, so each emulator is already built and tuned for optimal performance on the platform. Even better, BES requires you to map the buttons of your USB controller to the buttons of an SNES controller *only once for it to work with all of the BES emulators*, rather than requiring you to configure each emulator independently.

Within the full system image, the BES application lives in the *application root directory*: `/home/ubuntu/bes`. To make things easier for the end-user, all of the standard configuration files that control game configuration, controller button mapping, etc. are located inside of this directory and do not need to be touched. Instead, BES provides a web-based interface[1] for configuring button mappings, uploading and removing ROM files, and editing ROM descriptions as they appear in the BES front-end GUI.

If you do want to log into a running BES system, use the username "ubuntu" and password "temppwd". You can use the "ubuntu" account to `sudo` any administrative commands that you feel need to be performed on the system.

3.2 Configuring BES Using the Web Interface

Once the image has been copied onto the microSD card, you are almost ready to power up your BBB and run BES for the first time. Connect your BBB to your video display via the micro-HDMI port and place the BBB on your local network via the BBB's ethernet port. The BBB will receive an IP address for your local network via DHCP. Turn on your display and then connect your BBB's power supply to power up the board. You'll see the four blue LEDs on the

[1] The files for the web interface are located in the `/home/ubuntu/bes-config-node` directory.

BBB flash as it boots. Within about 10 to 20 seconds, the BES splash screen appears and then a welcome message is shown on your display.

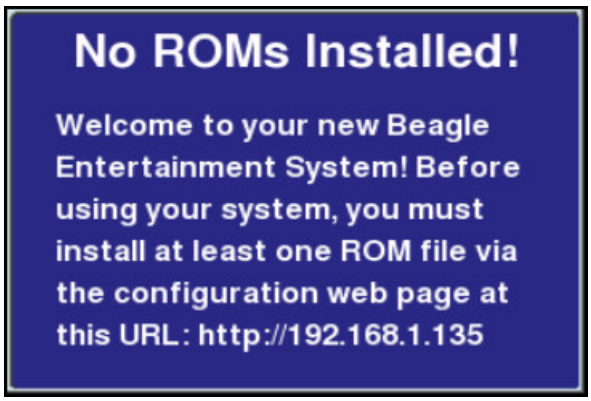

Figure 3.1: The welcome message for a fresh BES install. In this example, the IP address assigned to this BES via DHCP is 192.168.1.135.

The IP address assigned to BES is displayed on this screen. Using a web browser, browse to the URL shown in the welcome message to access BES's web-based configuration interface. Until at least one ROM is present on the system, this welcome message will be displayed on start-up and you will be unable to reach the BES game selection menu.

You must have BES connected to your network prior to powering up the system. The IP address is only assigned once during boot, so connecting BES to the network after boot will result in an IP address of "`No IP assigned`" to be shown.

The BES configuration web interface has three configuration screens, each of which can be accessed via the navigation tabs at the top of the interface. The leftmost tab allows you to add and remove ROMs to/from BES. The middle tab allows you to edit the description of any uploaded ROMs. The rightmost tab allows you to configure the button mapping of the USB controllers that you use with BES. Once you have made any changes to BES using the web interface, you must reboot the system for those changes to take effect. You can do this by simply cycling the power on your BBB (i.e. unplug and reconnect the power adapter to the BBB).

3.2.1 Adding and Removing ROMs

Click on the leftmost tab to change to the ROM Add/Remove interface page. This page allows you to add/remove ROMs and see how much available storage space remains on your microSD card to hold more ROMs, saved games, and save states. At the top of this page is a meter showing you the disk usage of your BES system. The meter tells you how much storage space is available and approximately what percentage of the microSD card is currently used.

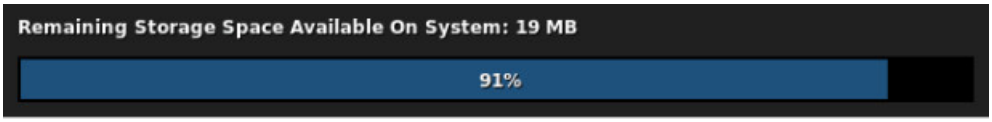

Figure 3.2: The BES storage space usage meter. This sample meter shows a 100 megabyte BES system that has 19 megabytes of storage space remaining. In an actual BES system, several gigabytes of storage are available for your use.

Below the storage space meter is the ROM add/remove interface.

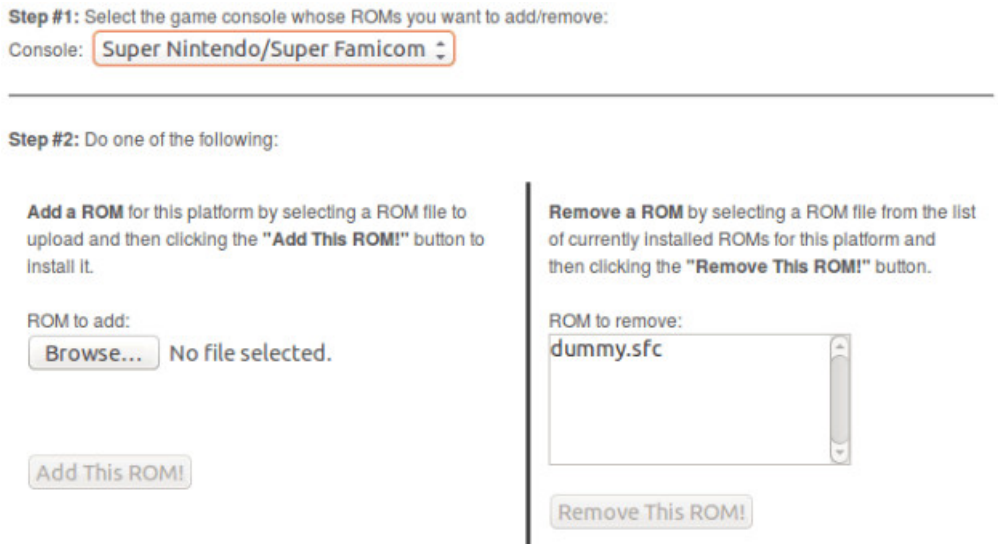

Figure 3.3: The BES ROM add/remove interface.

Whether adding or removing a ROM, the first step is to select the console whose ROMs you'll be working with from the drop-down menu:

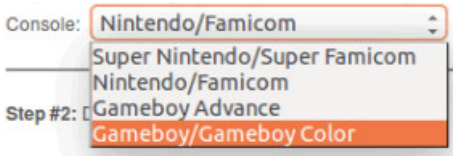

Figure 3.4: The platform selection drop-down menu.

To add a new ROM to the system:

- Click the "Browse" button to open a file dialog box. Select the ROM that you wish to add to the system.
- Browse to the ROM file on your system, and then select it.
- Click the "Add This ROM!" button to upload the ROM to BES.

If you browse to the location of a ROM file and don't see it, the ROM may have the wrong file extension. Make sure that SNES games have an `.sfc` or `.smc` extension, NES games have an `.nes` extension, Gameboy Advance games have a `.gba` exension, and Gameboy games have a `.gb` or `.gbc` extension.

To remove an existing ROM from the system:

- Select a ROM in the "ROM to remove" window by clicking on it.
- Click the "Remove This ROM!" button to remove the ROM from BES.

3.2.2 Customizing Game Menu Entries

Once one or more ROMs have been added to BES, BES will no longer display the "No ROMS Loaded" dialog screen on boot. Instead, it displays a game selection menu that shows all of the ROMs available on the system. By default, each ROM will have a very minimal entry within the menu. To configure these menu entries to be more informative, the web interface provides a game ROM information editing screen.

Click on the middle tab of the web interface to change to the Edit Game ROM interface web page. This page is completely optional, so you never even have to touch it to use BES. On this page, a preview pane is located on the right that shows you what the ROM's entry will look like in the menu with the current ROM information. All information fields that you can edit for the ROM information are located on the left. To begin editing, select the console whose games you wish to edit from the drop-down list. All ROMs installed for that console will appear in the "ROM Filename" list. Select a ROM from the list to begin editing.

Figure 3.5: The console selection drop-down list and ROM Filename list.

Once a ROM has been selected, its current information is displayed in both the fields on the left and the preview pane.

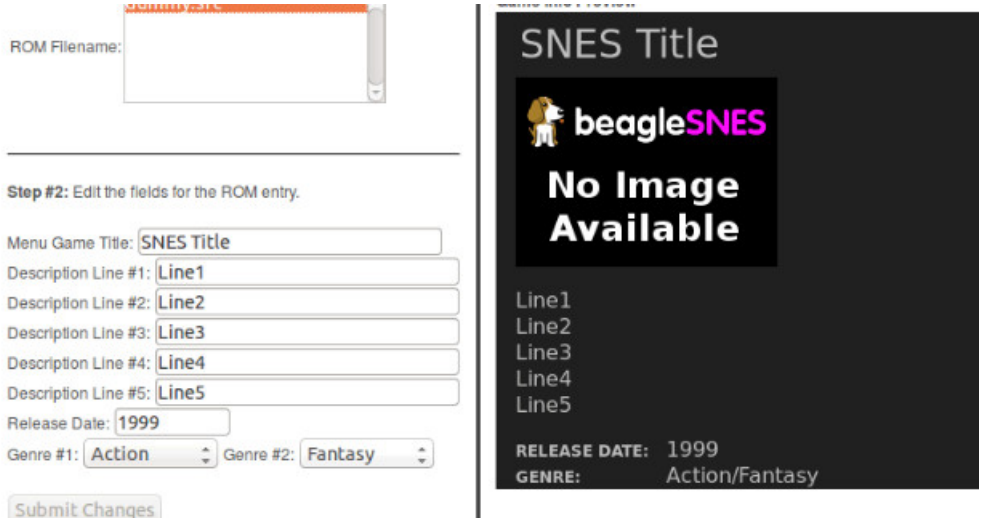

Figure 3.6: The ROM information editing fields and preview pane.

As you edit each field the preview pane is updated. Use this to make sure that any text that you enter fits on the screen and makes sense. In addition to the text fields, there are two genre drop-down lists. Each of these lists can be used to categorize the ROM into a genre categories (Action, RPG, Fantasy, Sci-Fi, etc.). If you do not wish to categorize the ROM, you can set the genre in both drop-down lists to "None".

3.2.3 Configuring Gamepad Button Mapping

There are a variety of USB human interface devices (HIDs) that are recognized as joysticks by the Linux kernel's USB subsystem. Not all of these devices are true joysticks (they might be gamepads, or even devices that looks nothing like a game controller), but they all fall within the

same category of "close enough in functionality to a joystick to be treated as such by the kernel". While these joysticks and gamepads can be used to provide input to BES, the physical buttons of each controller must first be mapped to the "logical" buttons inside BES that correspond to the buttons of the original SNES controller.

Figure 3.7: The original SNES controller. The L (left) and R (right) shoulder buttons are located on the top edge of the controller.

As a reference, the original SNES controller is shown in Figure 3.7. This controller has a directional pad that provides two "axes" for directional input (an up/down axis and a left-right axis). It also provides a total of eight buttons: the rectangular start and select buttons, four round input buttons for A/B/X/Y, and two shoulder buttons on the edge of the controller for L (left) and R (right).

Click on the rightmost tab of the web interface to change to the Configure Your Controllers interface web page. This page is used to specify the X-axis and Y-axis numbers of your controller, the “deadzone range” of analog sticks (when using them for the X and Y axes), axis inversion (if needed), and which of your controller buttons map to the SNES controller buttons. It can also be used to specify a gamepad button that, when pressed, will pause each emulator and bring up the BES pause menu. You can also specify a set of buttons that, when pressed together, will trigger a “pause combo” and bring up the BES pause menu.

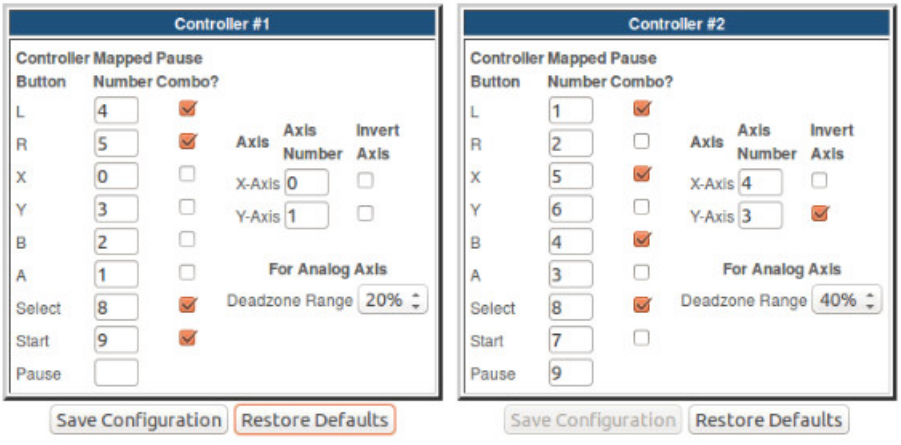

Figure 3.8: The controller button mapping configuration interface fields.

The best way to find out your button and axis numbers for your particular USB gamepad is to connect it to a Windows or Linux system and collect the information from there. Windows users can use the `joy.cpl` program to do this. Linux users can use `jstest-gtk` program to do this. The interface web page provides links to more information on installing and using these utilities.

If you can't spare an extra button to trigger the pause menu, you can map the pause to a combination of buttons. When this button combo is pressed together, the pause menu will trigger. Simply check the "Pause Combo?" box next to each button that you wish to include as part of the pause combo. For example, you can map the left and right trigger buttons and the start and select buttons together as a pause button combo by checking the box next to each of these buttons. Then, when all four are pressed simultaneously, the pause menu triggers.

You don't want the pause button combo to be a set of buttons that will interfere with gameplay, so selecting *at least* three to four buttons for the combo is advised. If you have enough buttons on your USB gamepad to allow for a dedicated pause button, though, it is advised that you do not specify any pause combo at all. Also, don't specify a pause button that overlaps with one of the regular SNES gamepad buttons!

BES also has some support for alternative inputs via GPIO and native SNES gamepad interfacing! These are a bit more advanced and require a little electronics know-how, but basic instructions on how to use these features are given in Chapter **??**.

3.3 Using BES

Once the system has been configured, and one or more ROMs have been uploaded, it is now ready for use. BES expects either one or two USB gamepads to be plugged in. The BBB has only one host USB port, so there are two options for using gamepads with the system. The first option is to plug a gamepad into the host USB port. This is the port for the "player one" gamepad, and no "player two" gamepad can be used. The second option is to plug a USB hub into the host USB port and then plug one or two gamepads into the hub. One port of the hub will be the "player one" port, and another will be for "player two". Whichever physical ports on the hub report as the first and second logical ports will be used as the "player one" and "player two" ports, respectively. *Any gamepad that is not plugged into one of these designated USB ports on the hub is ignored.*

3.3.1 Game Selection Menu

Assuming that you have successfully completed the steps of the adding ROMs and game descriptions to your BES system, it will now boot up to a complete menu of titles that are ready to be played. This game selection menu will *only* respond to the "player one" controller. Press up and down on the gamepad's direction-pad to change the currently selected title on the menu. Press either the "select" or "start" gamepad button to launch the currently selected title. Once a title has been launched, the menu will fade out and the emulator will begin execution.

Once you have launched a title and left the game selection menu, you must use the pause menu to return to the game selection menu to select a different game to play.

The web-based configuration interface is *only active while the game selection menu is up*. Once a game is selected and an emulator begins running, the web interface is shut down to give

more CPU to the emulation. If emulation ends and you return to the game selection menu, the web interface will be restarted.

Once the web interface is restarted, it can take about 10-15 seconds before it begins responding to web page requests for the interface. During this start-up period, the game selection menu can become a little sluggish.

3.3.2 Pause Menu

The pause menu can only be triggered while playing a game. Its purpose is to allow the user to load and save *snapshots* of the current game state, exit the current game and return to the game selection menu, and, of course, pause the game. Figure 3.9 shows the pause menu running under BeagleSNES, though all emulators within BES have a pause menu.

Figure 3.9: The pause menu. While in-game (left), triggering the pause menu will bring up a set of menu options (middle). If a snapshot for the current game has been previously saved, the "Load Snapshot" option will become active and a thumbnail will be shown in the "LAST SNAPSHOT" preview window.

There are three ways to trigger the pause menu:

- If the pause menu has been mapped to a single gamepad button via the web interface, press that button.
- If a pause button combination has been mapped via the web interface, press that button combo.
- If the pause menu has been mapped to a GPIO (for advanced users that are hacking on BES), trigger that GPIO.

Pressing up and down on the player 1 gamepad, or triggering the appropriate GPIOs, will navigate the pause menu. Likewise, using select/start or its GPIO equivalent will select a menu option. Selecting "Resume Game" will remove the pause menu and unpause the game. Also, pressing the pause button or button combo, or triggering the pause GPIO, a second time will also remove the pause menu and unpause the game.

The "Load Snapshot" and "Save Snapshot" menu options allow you to save the current state of the game and then load it to return to it at any time. This is useful for saving the progress of the current play session when you need to take a break and are not near a point in the game where progress can be saved. It is also useful for saving your game just prior to making that difficult jump or fighting a tough boss battle. If no snapshot for the current game exists, the "Load Snapshot" option will be grayed-out and unavailable. Only one snapshot can exist for each game. Once a snapshot is saved, the prior snapshot is overwritten. The screenshot associated with the snapshot for each game is also displayed on the game selection menu of BES.

The "Return to Game Selection" menu option exits the current game and return you to the game selection menu. Be sure to either save your current game from inside of the title you are playing (if such an option exists) or save a snapshot of your current game if you wish to return

to it later. Selecting this option is roughly the same as turning off the power on your gamining console: your current game in progress will be lost forever.

BEAGLE ENTERTAINMENT SYSTEM

4 — Troubleshooting

4.1 Overview

If you are unable to `dd` the full system image to your microSD card, or the system won't boot:

- Did you use `unxz` (or a Windows tool like 7-Zip[1]) to decompress the image prior to writing it to your microSD card? Decompressed images about 7.5 GB in size, while compressed images are about 400 MB.
- Is your download corrupt? Does the MD5 hash for the file match the hash listed for the file at SourceForge?
- Did you disable the write-protect switch on your microSD card? Look for the little switch on the side of your microSD-to-SD card adapter (if you are using one).
- Do you have permissions to write to the raw microSD card device file? Remember that you need to `sudo` the `dd` command (or run as root, which you *should not do*).
- Are any pre-existing partitions on the microSD card currently mounted? Unmount all partitions on the microSD card before performing the `dd`.

If you can't get your system to boot after writing the full system image to your microSD card:

- Are you using an external power supply (5VDC 2A)? The system won't boot when using only USB power.
- Did you disable the write-protect switch on your microSD card? The BES still needs to write out log files to the local file system.
- Is there any output coming from the debug serial or FTDI port? If there is nothing, `dd` the full file system to the microSD card again after you verify the MD5 hash of the downloaded file.
- Did you cleanly unmount the microSD card from the Linux system that you used to `dd` the image to the card? Some Linux systems will auto-mount the microSD card as soon as it has a valid file system on it. If you just pull the microSD card out when it is mounted, it is possible to damage the boot partition.

If you can see the boot splash screen, but then no IP is being shown or you can't connect to that IP:

[1] `http://www.7-zip.org/`

- Do you have your BBB connected via ethernet to a router that supplies IP addresses via DHCP? The BBB receives its IP via DHCP.
- Is your BBB connected when it is powered up? Make sure that the BBB is connected to your ethernet cable *before* powering it up.
- Is the PC you are using to connect to the BES IP on the same network as BES? Make sure that your PC is actually able to "see" BES (i.e. try to `ping` the IP address of BES).
- Have you given the web interface enough time to actually start running? It can take 10 to 15 seconds to get the web interface responding to requests when it is started upon reaching the game selection menu.
- If all else fails, send us a bug report and we'll check it out.

If the game selection menu appears, but it does not respond to your USB gamepad:

- Do you have the Tomee USB gamepad? If not, you may have to play with the button mappings via the web interface.
- Is your gamepad plugged into the correct USB port? Only a gamepad that is plugged into the "player one" USB port can control the menu selection screen.
- For the BBB, plugging an external USB hub into the host USB port can be very temperamental. If using an external hub, plug in the hub and all controllers prior to starting the system. Sometimes the external hub will stop responding if all controllers are removed from it. An unpowered external hub will work, but it makes the BBB far more sensitive and can create problems when hotplugging controllers.
- Verify that your gamepad is operating properly by testing it with another computer.

If the game selection menu starts, and you can launch a title, but then the screen goes blank and then "bounces" back to the menu:

- Make sure that the ROM isn't compressed via `zip`. All ROMs have to be uncompressed.
- Make sure that the ROM is valid by executing it under another emulator. BES only filters ROM by filename extension, so you may have something like a NES title named with an SNES extension by accident.

If you run into some other problem that is not covered here, please feel free to submit a bug report to Andrew at `hendersa@icculus.org`.

BEAGLE ENTERTAINMENT SYSTEM

5 — Hardware Hacking With BES

5.1 Overview

With the small physical size and flexibility of the BeagleBone family of boards, it is natural to consider using BES as the base software for creating a portable or standard gaming console. To this end, a lot of research has gone into building BES with a variety of hardware interfacing options. An overview of two possibilities is presented here: a handheld portable unit and a console with a very small form factor. Note that, while these prototypes have been tested in the past and do work, this is not intended to be a comprehensive guide to recreating them. Instead, this is a general set of guidelines that show what is possible should you decide to try creating projects like these using BES.

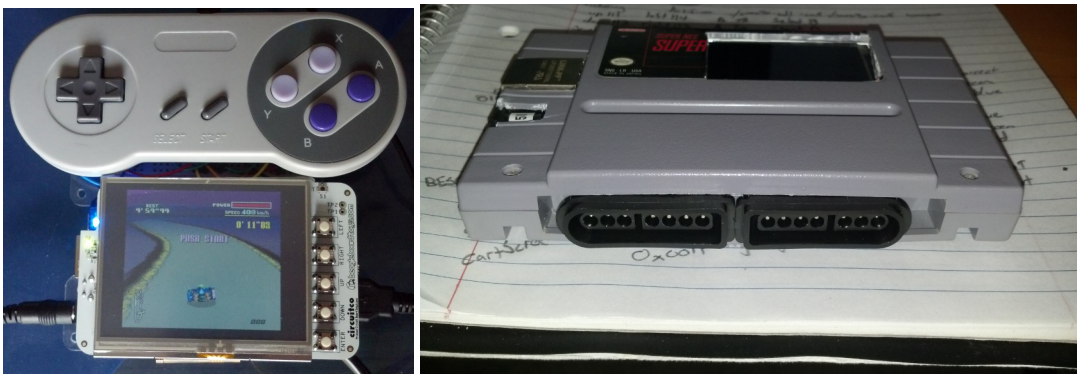

Figure 5.1: Prototypes for a portable BES system (left) and a console unit (right). The portable prototype still requires a USB gamepad and power via the +5VDC barrel connector, so it isn't fully "portable". The console prototype supports native SNES controllers and has a TFT LCD status display.

5.2 Making BES Portable

This portable variant is different from the base BBB target in the following ways:

- The LCD3 cape has a resolution of 320x240. This requires a rework of the BES front-end GUI. On the bright side, hardware-based OpenGL ES scaling makes this a simple task,

though some GUI aspects need to be reworked for readability.

- Additional audio hardware is needed. For the BBB HDMI target, a dummy CODEC in the kernel provides an ALSA interface to userspace. Audio data written to the dummy CODEC is passed to the HDMI framer chip in I2S format. The framer then packages up the audio data and sends it to the display via the HDMI protocol. When not using the HDMI framer chip, a different audio device must be provided.

5.2.1 Portable Video

The LCD3 cape, seen in Figure 5.2, provides the video display functionality for a portable target. It weighs approximately 70 grams and has a current consumption of 100mA. While it is possible to use a smaller display, or at least a display module that has a smaller footprint, the easy availability and pre-packaged nature of the LCD3 cape makes it appealing for quick prototyping without having to worry about device drivers.

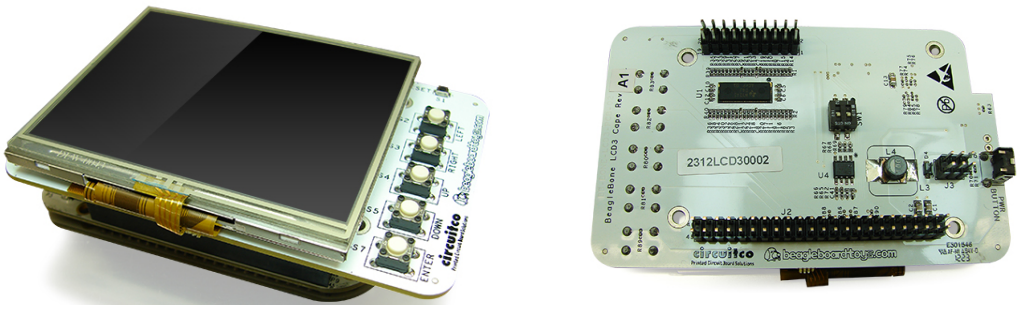

Figure 5.2: The LCD3 cape board made by CircuitCo. This 320x240 LCD has a touchscreen and five input buttons (left), and it plugs into the BBB's P8/P9 connectors via pins on the underside of the cape board (right). Photo credit: `http://elinux.org/CircuitCo:BeagleBone_LCD3`

Because the LCD3 cape is defined in the default Device Tree for the BeagleSNES kernel, it will be automatically detected and configured when it is connected to the BBB. The LCD3 uses the `TILCDC` driver in the kernel to provide a 16 BPP data bus between the BBB and the LCD3 cape. This is the same driver that is used to communicate with the built-in HDMI cape of the BBB, so userspace software need not be aware of the differences between communicating with the LCD3 cape versus communicating with the HDMI framer chip on the HDMI cape. As long as the 320x240 resolution is configured as a valid framebuffer resolution (by adding the appropriate framebuffer mode to the `/etc/fb.modes` file), BeagleSNES can simply request a 320x240 resolution and then adjust its rendering accordingly to use the LCD3 cape.

5.2.2 Portable Audio

The simplest solution for adding audio hardware to the prototype is to use a USB sound device. Such devices are inexpensive (usually around 5-10 USD), quite small (as seen in Figure 5.3), and readily supported via the USB sound device driver in the kernel. The configuration to enable this audio driver in the kernel is seen in Figure **??**.

The BBB only has a single USB port available. Plugging the USB audio device into it is convenient from an implementation standpoint, but it is inconvenient because the device will extended beyond the footprint of the BBB and LCD3 cape. Worse yet, a USB port is still needed for the USB gamepad. To resolve these issues, a small, 4-port USB hub is plugged into the single USB port and then used to provide USB to the gamepad and USB audio device. The hub, when its casing was removed, is so small that it fits into the space between the LCD3 cape and BBB when the cape is plugged into the BBB.

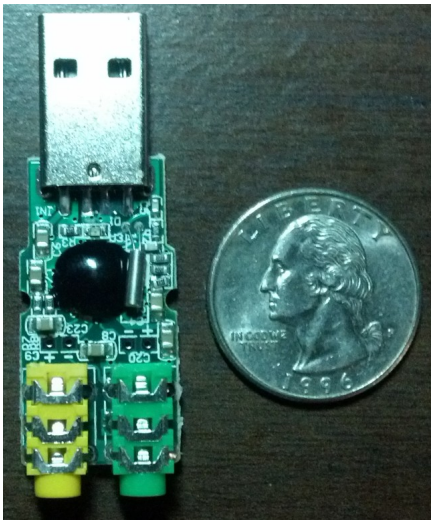

Figure 5.3: A USB audio device with its casing removed.

Figure 5.4 shows the positioning of the USB hub and the USB audio device when both components are positioned on top of the BBB. The USB audio device is soldered directly to one port of the USB hub, allowing the audio jacks to be positioned in a convenient location that isn't limited by the physical position of the USB ports on the hub. Both the hub and audio device are wrapped in electrical tape to avoid shorting and inadvertent electrical connections. Because the LCD3 cape only uses a portion of the pins on the P8 connector, the audio jacks are positioned on the connector, next to the ethernet jack. The assembled unit with all components is shown in Figure 5.5.

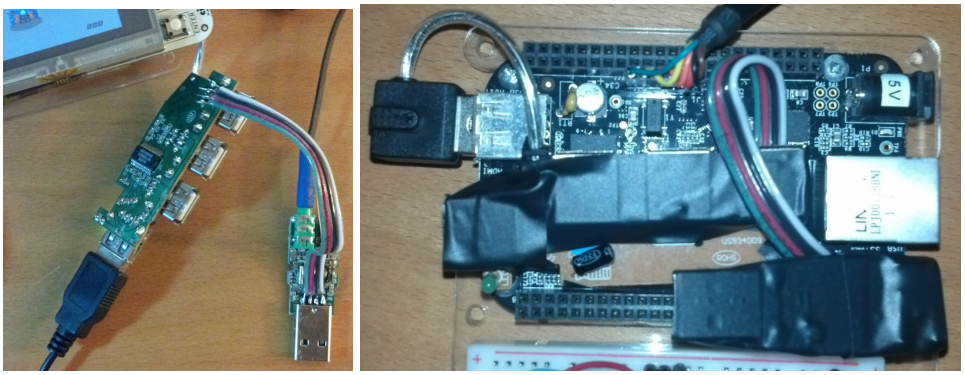

Figure 5.4: The USB hub and audio device. The audio device is soldered directly to a USB port on the hub (left). Both the hub and device are wrapped in electrical tape and positioned on the BBB.

The CircuitCo Audio Cape requires the use of GPIO3[19], which is also used by the LCD3 cape. So, as the moment, there is no way of using the two together. A copy of the device tree overlay for the audio cape has been added to `/lib/firmware` so that you can experiment with the audio cape, but it won't just "plug-and-play" like most capes will. You will have to manually load the overlay for the audio cape to use it.

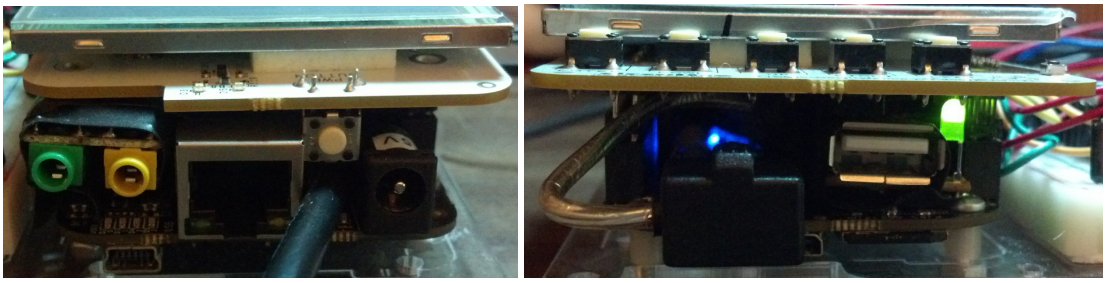

Figure 5.5: The assembled prototype. The edge of the unit with the power connector (left) exposes the audio jacks and the FTDI debug cable. The edge of the unit with the BBB USB port (right) exposes the edge of the USB hub and the USB port exposed by the hub. The slight "tilt" of the LCD3 cape is due to the height of the FTDI debug cable preventing one side of the cape from being completely inserted.

5.2.3 GPIO Input

While requiring the user to use a USB gamepad for input is convenient for the developer, it isn't convenient for the user. A proper implementation of a portable unit would use GPIO buttons for input and eliminate the need for the USB hub completely. Luckily, BES supports GPIO input. This allows for a more elegant design for a portable platform, since a custom controller can now be built around the LCD and BBB to minimize the unit's size and weight.

By default, the GPIO support for BES on the BBB uses all of the 13 pins on P8.7 through P8.19. These pins are configured as input pins, and BES maps these pins to SNES controller events. In addition, pin P8.2 is used as a ground, and pin P9.3 is used to provide 3.3V. When 3.3V is applied to a GPIO pin, a key down event is placed into the internal application event queue. When the GPIO pin goes to ground, a key up event is triggered in the same fashion. Figure 5.6 shows an example implementation of GPIO input for BES. Each pin has an external 10k ohm resistor in a pull-down configuration.

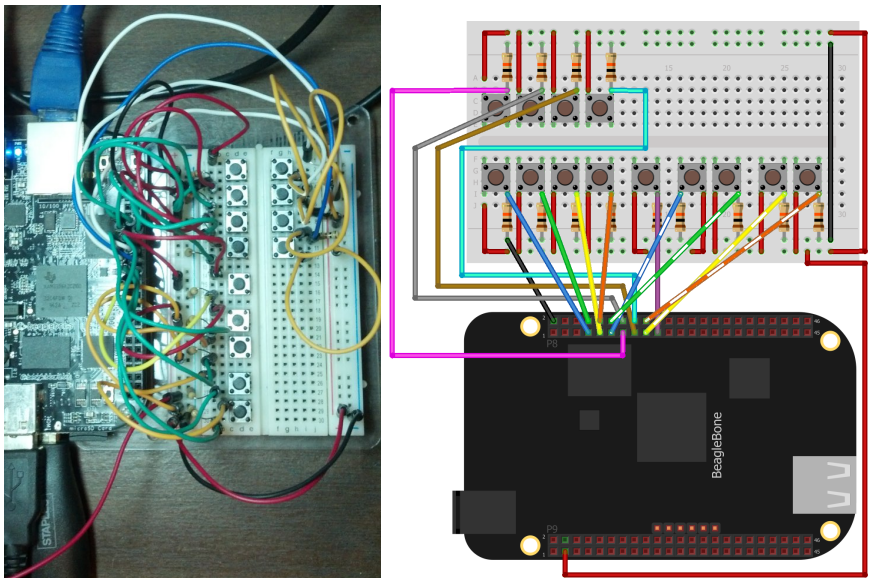

Figure 5.6: Input via GPIO. Thirteen pins are in use in the contiguous block from P8.7 through P8.19. Each pin uses a pull-down resistor configuration. These pins implement the directional pad (4), A/B/X/Y/L/R buttons (6), start/select buttons (2), and a pause button.

While most of these pins can be easily accessed while the LCD3 cape is attached, pin P9.3 is used to supply power to the LCD3. To connect to this pin, either put an intermediary "prototyping" cape in place that will give you access to the pin or snap a connector onto the P9.3 pin of the LCD3 cape to make a connection. The prototype in Figure 5.6 used a logic analyzer probe connector to grab onto the P9.3 pin of the LCD3 cape.

Always be very, very careful when working with pins on the BBB that supply voltage. While pins P9.3 and P9.4 supply 3.3V, pins P9.5 and P9.6 supply 5V. Connecting a 5V line to a GPIO is a sure-fire way to ruin your BBB, so always exercise caution!

5.2.4 Other Considerations

While the prototype is a good start, there is still work to be done in making a true "portable" system:

- Power is a consideration. How much battery life will you get if the emulator is maxing out the CPU? You'll still need a 5VDC, 2A power supply to power the board with the AM3359 running at maximum capacity. How heavy will this battery be? The design of the (now discontinued) CircuitCo battery cape[1] might be a good reference point to start designing from.

Ultimately, it will come down to how fancy you wish to make your portable unit design and how much money you'd like to spend prototyping it. The sky is the limit!

5.3 Making A BES Console

Unlike the extensive effort required for a BES portable, a BES console is much quicker to get together and working. This console variant is actually is running in the base BES image, and by wiring additional components into your BBB, these features will "just work". It provides the following features:

- Input to the system comes via native SNES gamepad interfacing. The BBB PRU subsystem performs this interfacing by "bitbanging" the SNES protocol.
- A status display is provided using a combination of SPI and GPIO interfacing.

5.3.1 Prototype overview

There is a *lot* of material to cover to fully explain this prototype, and since I'd like to get this software released sometime this year, I'm not going to go into it in a great level of detail. The quick summary is that the console configuration uses an SNES catridge shell and two Adafruit components to create the console. The Adafruit ILI9340 TFT LCD[2] and Adafruit BSS138 four-channel logic-level conversion board[3] are used, as are Raphnet SNES gamepad controller connectors[4].

Until I get around to documenting this, please contact Andrew at `hendersa@icculus.org` for more details.

[1] `http://elinux.org/CircuitCo:BeagleBone_Battery`
[2] `https://www.adafruit.com/products/1480`
[3] `https://www.adafruit.com/products/757`
[4] `http://www.raphnet-tech.com/products/snes_controller_connector/index.php`

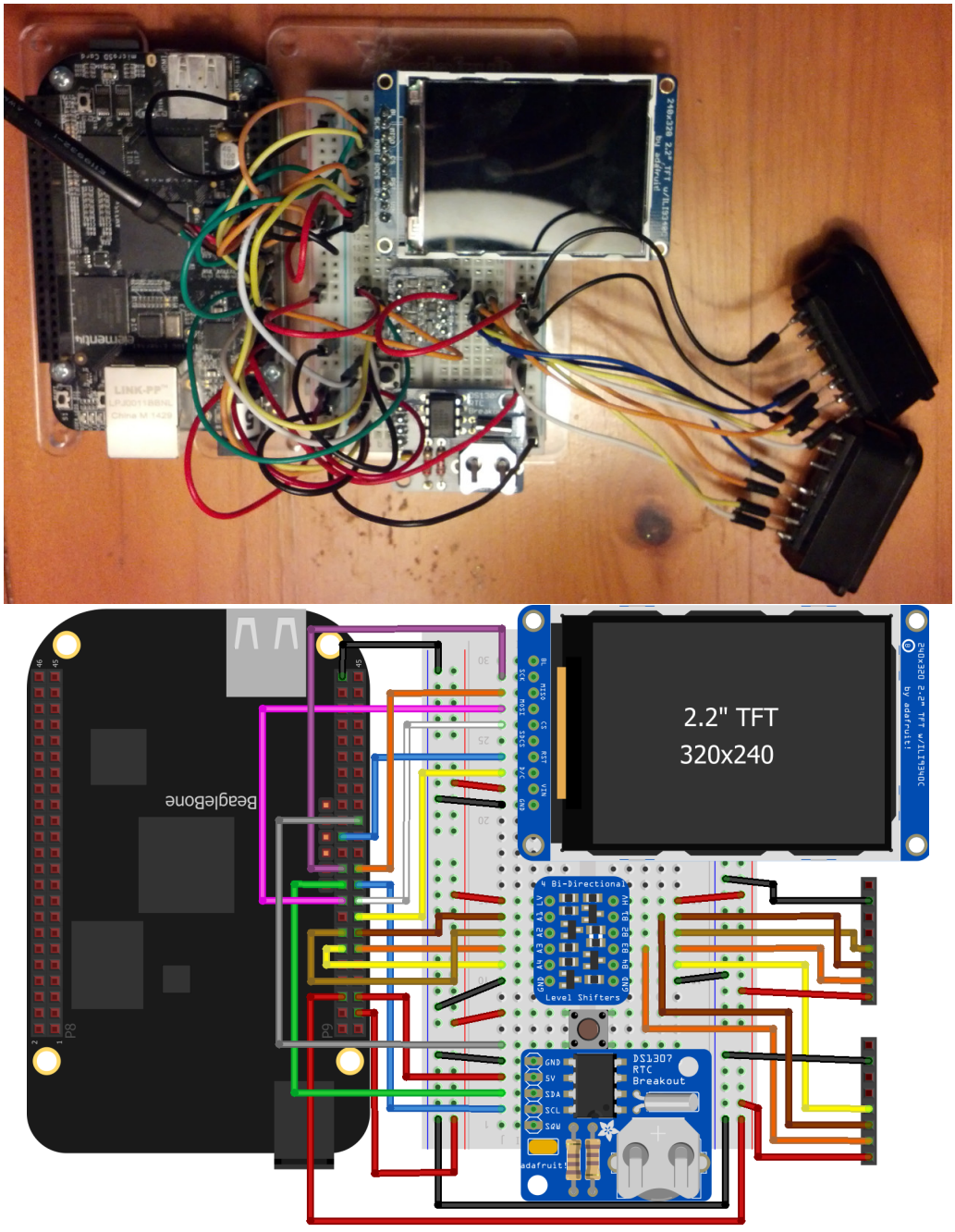

Figure 5.7: BES console prototype. Note that a DS1307 RTC is included in this design, though support for the RTC is not in the default BES support.

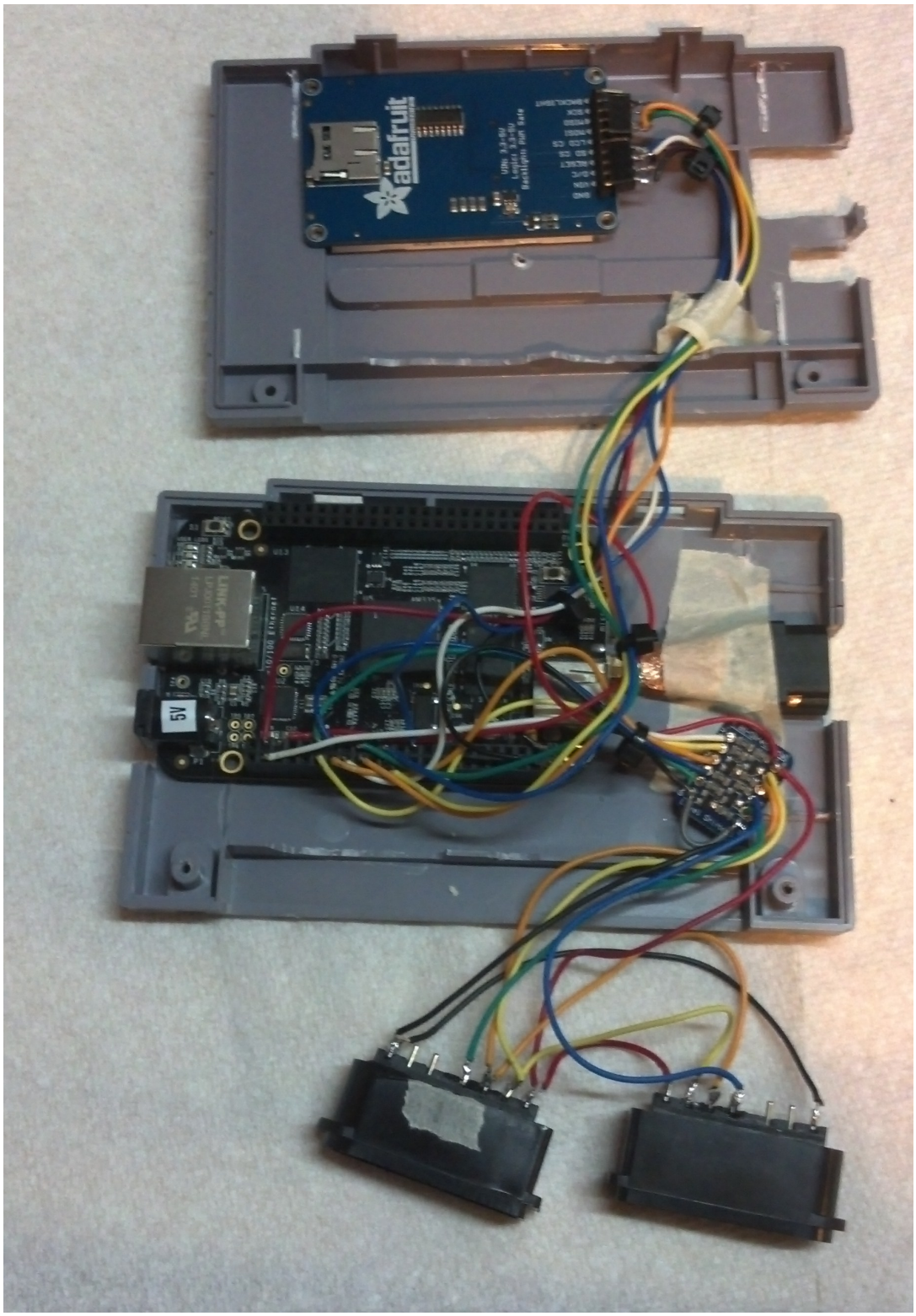

Figure 5.8: BES console prototype wiring and positioning within the SNES catridge shell. Note that a microHDMI-to-HDMI adapter has been used to not only convert the BBB's HDMI output to a full-sized HDMI connector, but it also pins the BBB in place against within the cartridge.

BEAGLE ENTERTAINMENT SYSTEM

6 — Acknowledgements

6.1 General Acknowledgements

While a project as complex as BES can be organized and put together by one person, such an effort is only made possible by leveraging the hard work that others have already done. The following people have all contributed to the pieces that make up the foundation that BES is built upon:

- The team at `beagleboard.org`, who have developed the BeagleBoard family of hardware platforms. Their efforts over the past several years have led to a variety of affordable, open-source hardware that is helping to encourage education and experience in embedded systems.
- Canonical (`http://www.canonical.com`), for their work on Ubuntu, which forms the base OS and file system environment of BES.
- Robert C. Nelson (`http://www.rcn-ee.com`), whose kernel work has formed the basis of BES's kernel. His wonderful Git repository of BeagleBoard kernel patches, as well as his numerous postings on various mailing lists and message boards, has provided a wealth of knowledge regarding the BBB platform and has greatly simplified the process in getting OpenGLES working on the BBB.
- The SDL library team (`http://www.libsdl.org`), for their on-going work in providing a straight-forward interface to the low-level multimedia systems of both desktop and embedded platforms.
- The SNES9X team (`http://www.snes9x.com`), for their cross-platform SNES emulator.
- The Nestopia team (`http://nestopia.sourceforge.net/`), for their cross-platform NES emulator.
- The VBA-M team (`https://sourceforge.net/projects/vbam/`), for their cross-platform Gameboy/GBA emulator.

There is more to BES than just code. The following people have also provided assistance and support during the development of the project:

- Dave Vedder at changeMode design (`http://www.chmoddesign.com`), who developed the layout and look of the game selection menu. If you need a GUI designed, I highly recommend him.
- Internet Janitor and Suspicious Dish of the Something Awful forums, who have provided useful feedback pertaining to the game selection menu's fonts and animations.

- Mathias Legrand, creator of the Legrand Orange Book LaTeX template[1] that this document was created with.
- Elijah Hall (`http://mrkittie.newgrounds.com`),
- ... Genraltweet (`http://genraltweet.newgrounds.com`),
- ... conorstrejcek (`http://conorstrejcek.newgrounds.com`),
- ... and MarioMan94 (`http://marioman94.newgrounds.com`), the creators of several pieces of music that have served as the background audio for the BeagleSNES release video trailers.
- Robert Beagley, who has lent his surname to the cause and who has also greatly enlightened the world at large with his many philosophical musings on bacon.
- The EECS department at Syracuse University (`http://www.syr.edu`), who gave guidance and support to the BeagleSNES project during its initial development as a graduate course project.

Finally, a big thank you to Nintendo®for developing the NES, SNES, Gameboy, and GBA hardware platforms.

[1] `http://www.latextemplates.com/template/the-legrand-orange-book`