adafruit learning system

Retro Gaming with Raspberry Pi

Created by Phillip Burgess

Last updated on 2013-08-09 06:30:25 PM EDT

Guide Contents

Overview

I'm a child of the 1980s. Miami Vice! Skinny ties! Big hair! Honest, I had hair then...and every town had *at least* one good video game arcade.

Thanks to the super affordable Raspberry Pi and some clever software, anyone can re-create the classic arcade experience at home. Adafruit brings the genuine "clicky" arcade controls, you bring the game files and a little crafting skill to build it.

Classic game emulation used to require a well-speced PC and specialized adapters for the controls, so it's exciting to see this trickle down to a $40 system.

Raspberry Pi Setup

The Raspberry Pi reads software off a regular SD memory card (or a microSD card with adapter). You'll need one that's **4GB or larger.** We'll start by loading up the Linux operating system on this card.

There are many "flavors" of Linux available for the Raspberry Pi. For this project, **we recommend starting with the official *Raspbian* distribution,** which can be downloaded here:

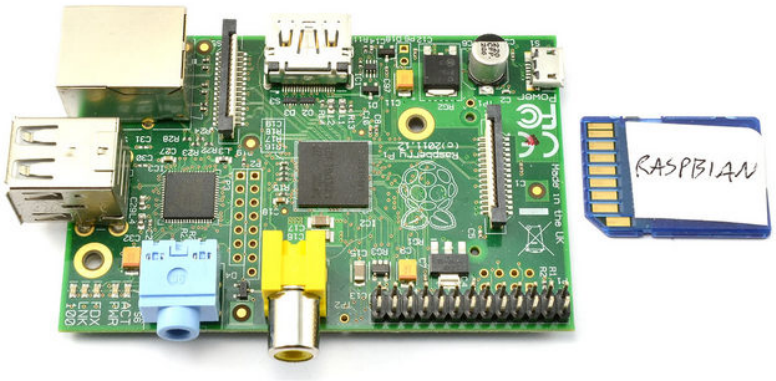
- Raspbian "Wheezy" Downloads (http://adafru.it/aMY)

This file is about **500 megabytes** and will take some time to download. While that's working, there are some other things you can do:

- Insert the SD card in the card reader, connect to your computer and format the card as a FAT32 (MS-DOS) filesystem.
- Confirm that you have all the other parts needed for your Raspberry Pi: power supply, USB microB cable (for power), HDMI or composite monitor, USB keyboard, mouse and a network connection (a wired Ethernet connection is easier, but WiFi with a supported USB adapter is possible with a few additional steps).
- Prepare a Raspberry Pi case, if you have one (peel plastic from acrylic, etc.). Tutorial link (http://adafru.it/c6i).

After downloading the software, uncompress the ZIP file in preparation for the next step...Here's a tutorial explaining how to install the downloaded software onto the SD card (http://adafru.it/aYV), with links to nice GUI apps for Windows and Mac. The first couple of pages can be skipped, as we're already downloading the right software for this project. Since that tutorial was written, another option has become available for Mac: RPi-sd card builder (http://adafru.it/aYW) — this one's even simpler, with no Terminal commands required.

> Use Raspbian 2013-02-09 or later for this project. Occidentalis 0.2 requires very lengthy updates and patches to work.

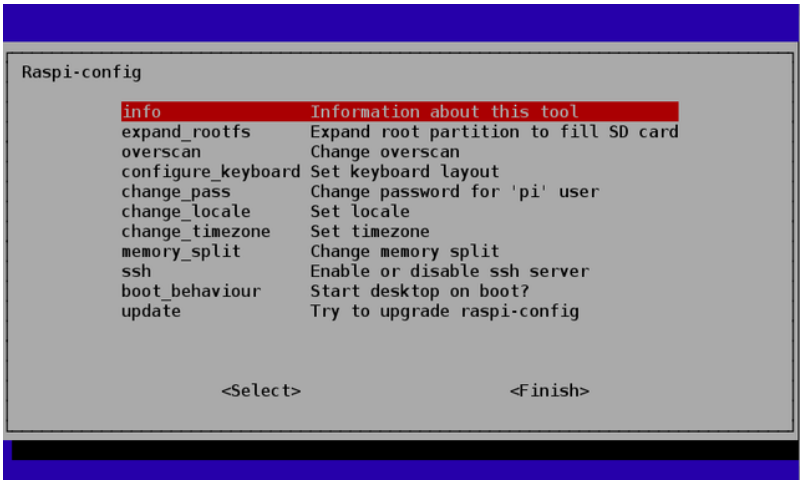


OS Install and First-Time Configuration

1. Connect a monitor and a USB keyboard to the Raspberry Pi.
2. Insert the SD card containing the Raspbian Linux software.
3. Connect a "Micro B" USB cable to the power connector on the Raspberry Pi.
4. Plug the other end of the USB cable into a power source: a mobile phone charger, a powered USB hub, or simply a USB port on your computer.

The Raspberry Pi should now boot, and you'll see the monitor fill with lots of "Unix stuff."

> Linux can be daunting to the uninitiated. Don't worry about messing something up… if all else fails, you can re-format the SD card and begin again.



When the system is started for the first time, the **raspi-config** utility runs automatically. You must select the following options:

- Expand root partition (this lets us use the full capacity the SD card).
- Configure keyboard ("Generic 105-key (Intl) PC" is the default — for the US and most other countries, you'll want to select an appropriate keyboard layout such as "Generic 104-key PC").
- Set timezone.

These steps are optional but recommended:

- If using an HDMI monitor, disable overscan.
- Change hostname. I called my system "retro" to distinguish it from other Raspberry Pis already on the network.
- Change password for "pi" user.
- Change memory split. Allocate at least 64 MB to the GPU.
- Enable SSH server (for administration access to the Raspberry Pi over the network).

These options should NOT be selected:

- Start desktop on boot (we'll run X11 manually when we need it).

A more in-depth tutorial on Raspi-config is available here (http://adafru.it/aYX).

After configuring these options, select “Finish” and reboot the system when prompted.

The first reboot will take a couple minutes while the filesystem is expanded for the SD card.

> Newer versions of raspi-config have additional options such as overclocking. Even moderate overclocking runs the risk of corrupting the SD card. For this project we recommend leaving the Raspberry Pi at the default speed.

Network Configuration

If you have a wired Ethernet connection, nothing special needs to be done here. Your Raspberry Pi should already be able to access the internet.

For wireless networking with a compatible USB-to-WiFi adapter:

- Plug in the USB-to-WiFi adapter (a powered USB hub is recommended).
- Log in as user “pi” (using the password you set up during raspi-config, or “raspberry” as the default).
- Type “startx” to begin a desktop session.
- Double-click the WiFi Config app on the desktop, select “Add” from the “Manage Networks” tab and enter the information for your wireless network.

> This currently works only with “broadcast” SSID networks. Hidden networks cannot be accessed.

Mac users: you can optionally install the “netatalk” package to enable the Apple file sharing protocol on the Raspberry Pi. This simplifies moving files to the system.

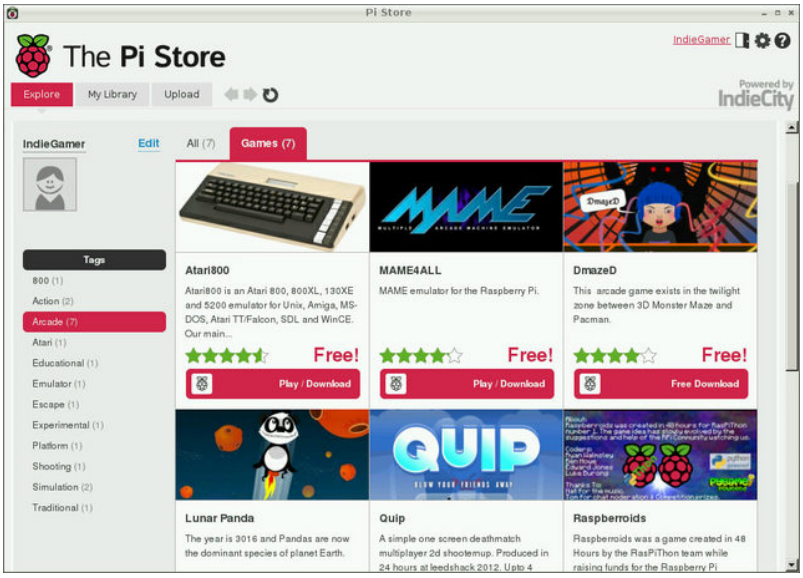
From the console or a terminal window on the Raspberry Pi, type:

```
sudo apt-get install netatalk
```

This also makes remote administration easier. From a Terminal window on the Mac, type “ssh pi@retro.local” — a numeric IP address isn’t needed.

Install Emulator(s)

We've tested a number of different game emulators and installation methods, and the easiest by far is to use the **Pi Store** application, which appears as an icon on your X11 desktop (type "startx" if you're currently running in console mode).



If this is your first time using the Pi Store, select the "Log In" link, then the "Register" button. You'll be asked for an email address and a password, but no other personal information is collected.

There are currently four good classic gaming emulators available through the store, all of them free downloads:

- **MAME4ALL** — emulates a number of classic arcade games.
- **PiSNES** — Super NES emulator.
- **pcsx_reARMed** — PlayStation 1.
- **Atari800** — Atari 8-bit computers (800, XL, XE, etc.)

All of these contain the emulation software only, no games are included. **You will need to track down your own ROMs or disk images to complete the process.** The descriptions for each package explain where to install the ROM files; there's a different directory for each one.

Most of these emulators can run either in X11 or from the console (except for Atari800 — console only). Performance is much smoother in console mode, so we recommend logging out of your X11 session after the programs are installed, then launching them from the command line. You will find the executable files in sub-directories of /usr/local/bin/indiecity/InstalledApps. For example, to run MAME4ALL, type:

```
/usr/local/bin/indiecity/InstalledApps/mame4all_pi/Full/mame
```

You can edit the file **.bashrc** in your home directory to set up simpler aliases for this, or add the directories to your PATH.

Before proceeding with the next step — acquiring arcade controls and investing the "sweat equity" of wiring these up and making a case — **confirm first that you can get the games that interest you working.** For reasons of performance on the Raspberry Pi's limited hardware, some of these emulators are based on earlier, simpler code releases and aren't compatible with the widest selection of games.

More downloads!

We haven't tried RetroPie but it might also work! (http://adafru.it/cfK)

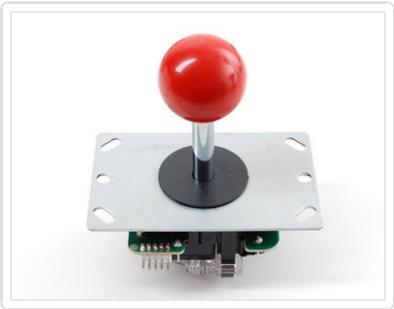
Adding Arcade Controls

At this point you should have one (or more) games playable from the keyboard. The next step is adding arcade-style controls…

Now things take a creative turn. There's no One Right Way to arrange controls that can cover every game. Instead, pick a few favorites and devise a layout that handles the most frequently-used inputs well. For everything else, you can still use a keyboard.

You'll also need to build your control panel using the **materials and tools best suited to your own skills**. I'm fortunate to have access to a laser cutter that can work with acrylic, but that's a tall order for most. Scrap plywood or a metal project box are viable materials (a cigar box works great too), while a drill, hole saw, Dremel tool or wood rasp are all reasonable tools for making holes. Improvise!

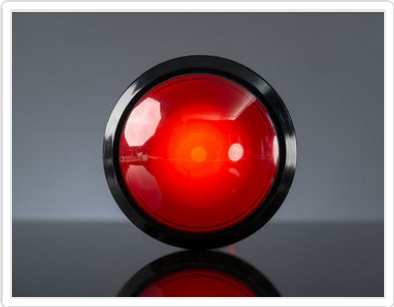
Maybe you have a supply of buttons on hand. If not, we have a nice assortment of genuine arcade controls in the shop:



Our Small Arcade Joystick (http://adafru.it/480) is the gamer's equivalent of the IBM Model M keyboard — clicky and built like a tank!

This is an 8-way "digital" joystick. A majority of classic games are designed for this type of stick, and the Raspberry Pi can't read proportional analog sticks directly.

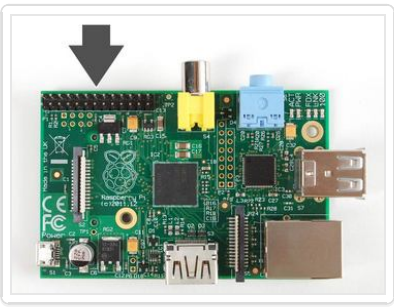
Our 30mm Arcade Button (http://adafru.it/471) is available in six different colors and is similarly industrial-grade. But any momentary push-type button ("normally open" contacts) will work fine.



Large (http://adafru.it/1192) (60mm) and Massive (http://adafru.it/1187) (100mm) arcade buttons come in five colors and are irresistible! These bigger buttons are typically seen on quiz games rather than fast-twitch shooters.

A classic Atari or Commodore joystick can also be used. Most later gaming consoles used serial protocols for their controls…unfortunately, those won't work here. USB game pads are another viable option.

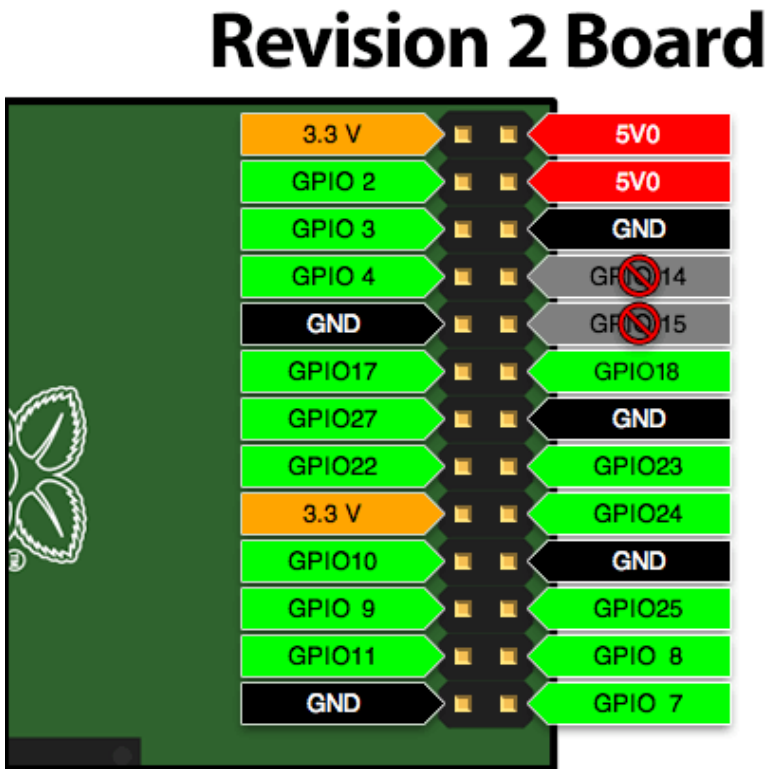
> We offer a variety of arcade controls, but we don't sell COMMON SENSE. When drilling and cutting, take precautions such as wearing SAFETY GLASSES, always cut AWAY from yourself, etc.



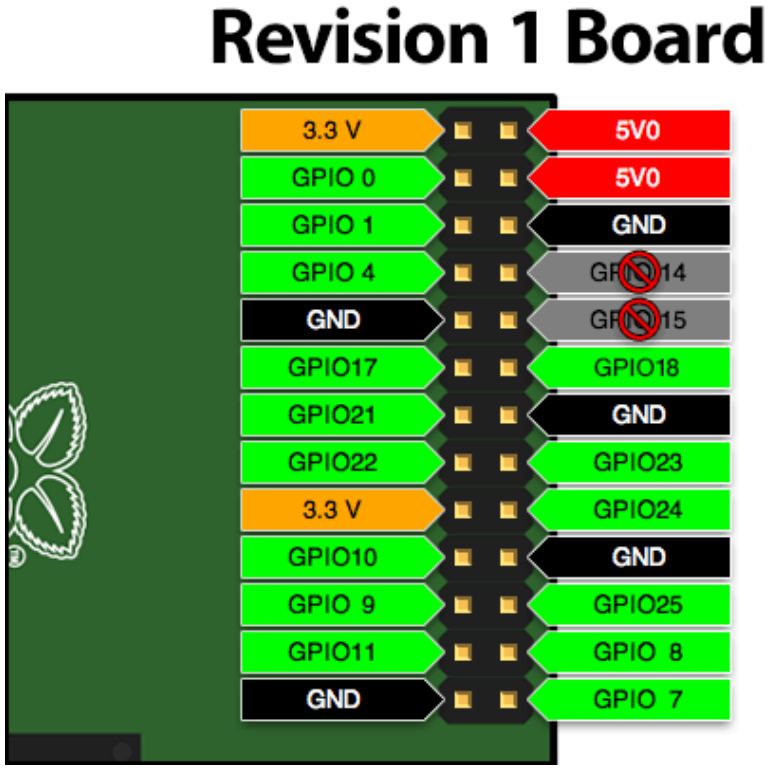
The controls will be wired to the 26-pin GPIO (general-purpose input/output) header on the Raspberry Pi board.

Each pin on this header has a unique "GPIO number" distinct from the order of the pins on the header. Buttons and a joystick (each of 4 directions) will connect between any available GPIO pin and a ground (GND) connection.

Most Raspberry Pi boards follow the newer "Revision 2" pin arrangement:



If you're using an early Model B board (easy to spot — there's no mounting holes), a few of the GPIO pin numbers are different:



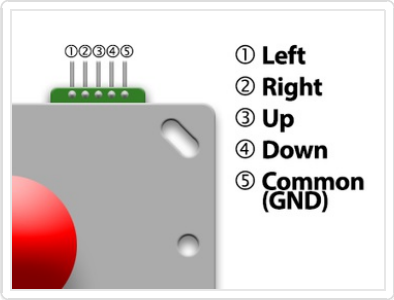
Notice that GPIO pins 14 and 15 are marked off-limits. These are set up as a serial port by default. There's a way to disable this, but it's somewhat annoying and there are plenty of other pins to work with, much simpler.

Revision 2 boards have four extra GPIO pins (28–31) on an adjacent header, but this requires some soldering. If you *desperately need* the extra inputs, power users can find that information in the Embedded Linux Wiki (http://adafru.it/cbG).

There are only five ground pins available on the header, but some gamers will need more controls than that. One of our small Perma-Proto boards (http://adafru.it/1171) can be used to provide a single large "ground rail" where one side of all the buttons can be connected. Alternately, if you have extra unused GPIO pins and just need a couple extra ground connections, we'll show a software work-around for this.

The processor in the Raspberry Pi has the ability to turn on 'internal' pullups so you do not need external pull-up resistors for the buttons or arcade controller

Just as the layout and build technique of the control panel requires creative interpretation, so too will you need to decide on your own wiring methodology. We have some parts that can help. Aside from the aforementioned Perma-Proto board, there are quick-connect wires (http://adafru.it/1152) that work with the buttons and jumper wires (http://adafru.it/793) in various lengths that can be used with a joystick or plug directly into the GPIO pins (without a Perma-Proto board). Options abound! You'll likely need some combination of these, and may need to solder some connections.



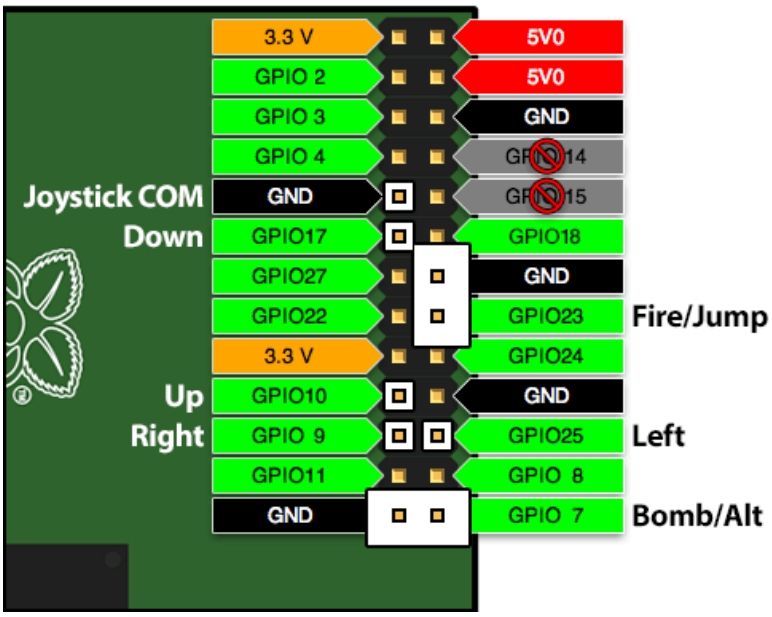
Here's a pinout diagram for our arcade stick. Only one wire needs to go to GND, then each of the other four goes to a different GPIO pin.

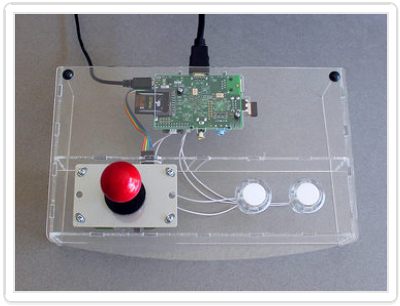
These directions apply when the stick is oriented **with the header at the top**. It's fine to install the stick in a different orientation, you'll just need to adapt the wiring connections to match.

The plugs on the ends of the Quick Connect Wires tend to block adjacent pins on the GPIO header. You can trim these down a bit using an X-Acto knife or Dremel tool, or cut the plugs off and solder the wires to a Perma Proto board, or simply plan out your wiring to avoid nearby pins.

Here's a **no-soldering** wiring setup we use with one joystick and two buttons, using 5 female-to-female jumpers for the joystick and two unmodified quick connects for the buttons. It's a pretty basic layout, but sufficient to accommodate quite a few classic games. For the remaining seldom-used functions (coin insert, start game), a regular USB keyboard is kept nearby.

If you dont mind doing some soldering, you can connect to the other pins - just make sure one pin of each button connects to a GPIO and the other button pin connects to one of the ground pins. You may need to edit/recompile our 'helper'

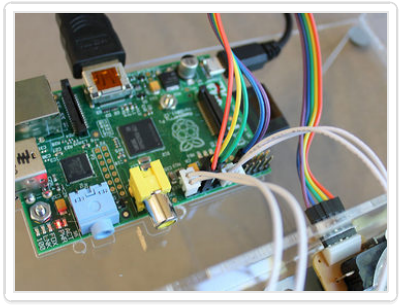




This is an example of a case I made from acrylic, but any workable material will do. **A cigar box makes a great fit!**

The case should be at least 50mm (2 in.) deep to accommodate the buttons and solderless connectors.

The shelf at the top of the case holds a small USB keyboard. This is a catch-all for seldom-used functions (e.g. game start). Only the essential controls were assigned to arcade buttons. Add others if you like!



Here's the joystick and two buttons plugged directly into the Raspberry Pi GPIO header as previously described.

One Last Piece...

Our “retrogame” utility is the software glue that joins these keyboard-driven games to our GPIO-connected controls.

Download retrogame from GitHub. (http://adafru.it/cbI) Select the “download ZIP” button and uncompress this file on your Raspberry Pi.

A pre-built version of retrogame is included in the ZIP file. If using the joystick and button layout described above, this program can be used as-is. Otherwise, the C source code needs some editing, and you’ll need to compile it (a Makefile is provided) after making changes. The GPIO pin to keyboard remapping is currently handled with a table in the source...this should be edited to match your unique wiring setup.

```
cd Adafruit-Retrogame
nano retrogame.c
```

(Of course you can substitute your editor of preference there — “nano” is easier for beginners.)

Starting around line 77 you’ll see a table resembling this:

```
struct {
	int pin;
	int key;
} io[] = {
//	  Input	  Output (from /usr/include/linux/input.h)
	{ 25,	KEY_LEFT	},
	{  9,	KEY_RIGHT	},
	{ 10,	KEY_UP		},
	{ 17,	KEY_DOWN	},
	{ 23,	KEY_LEFTCTRL	},
	{  7,	KEY_LEFTALT	}
};
```

Each line in this table contains two elements. The first is a GPIO pin number (where a button or one direction from a joystick is attached), the second is the corresponding key code to be generated by this control. A list of valid key code names can be found in the file /usr/include/linux/input.h starting around line 178. Remember to enclose each pin/key pair in {curly braces} with a comma between them.

By default, the code is set up to match our example “no solder” controller wiring. This also works directly with the standard MAME controls. So you might not need to change anything; the pre-built version may have you covered.

If you need an extra ground pin (and have extra GPIO pins available that you’re not using for controls), set the key code to GND instead.

Write your changes to the file and exit the editor, then type:

```
make
```

This should build the executable **retrogame** utility. If you instead get an error message, there's a problem in the edited table — most likely a missing curly brace, comma or semicolon.

But wait...we're not ready to run yet! Retrogame requires the *uinput* kernel module. This is already present on the system but isn't enabled by default. For testing, you can type:

```
sudo modprobe uinput
```

To make this **persistent between reboots**, append a line to /etc/modules:

```
sudo sh -c 'echo uinput >> /etc/modules'
```

Now we're in good shape to test it!

Retrogame needs to be run as root, i.e.:

```
sudo ./retrogame
```

Give it a try. If it seems to be working, press control+C to stop the program and we'll then set up the system to launch this automatically in the background at startup.

```
sudo nano /etc/rc.local
```

Before the final "exit 0" line, insert this line:

```
/home/pi/Adafruit-Retrogame/retrogame &
```

If you placed the software in a different location, this line should be changed accordingly. "sudo" isn't necessary here because the rc.local script is already run as root.

Reboot the system to test the startup function:

```
sudo reboot
```

The software will now be patiently waiting in the background, ready for use with any emulators.

Each emulator will have its own method for configuring keyboard input. Set them up so the keys match your controller outputs. Up/down/left/right from the arrow keys is a pretty common default among these programs, but the rest will usually require some tweaking.