Why CircuitPython?

CircuitPython is a variant of MicroPython, a very small version of Python that can fit on a microcontroller. Python is the fastest-growing programming language. It's taught in schools, used in coding bootcamps, popular with scientists and of course programmers at companies use it a lot!

CircuitPython adds the Circuit part to the Python part. Letting you program in Python and talk to Circuitry like sensors, motors, and LEDs!

CircuitPython Libraries on Linux & Raspberry Pi

The next obvious step is to bring CircuitPython **back** to 'desktop Python'. We've got tons of projects, libraries and example code for CircuitPython on microcontrollers, and thanks to the flexibility and power of Python it's pretty easy to get that code working with micro-computers like Raspberry Pi or other 'Linux with GPIO pins available' single board computers.

We are not running the CircuitPython interpreter itself on the Linux machine. But we *are* running Python code written to use the CircuitPython hardware API (busio.I2C, busio.SPI, etc.)

We'll use a special library called **[adafruit\_blinka](https://pypi.org/project/Adafruit-Blinka/)** ([named after Blinka, the CircuitPython mascot](https://www.adafruit.com/?q=blinka)) to provide the layer that translates the CircuitPython hardware API to whatever library the Linux board provides. For example, on Raspberry Pi we use the python [RPi.GPIO](https://pypi.org/project/RPi.GPIO/) library. For any I2C interfacing we'll use ioctl messages to the /dev/i2c device. For SPI we'll use the spidev python library, etc. These details don't matter so much because they all happen underneath the **adafruit\_blinka** layer.

The upshot is that any code we have for CircuitPython will be instantly and easily runnable on Linux computers like Raspberry Pi.

*In particular*, we'll be able to use all of our device drivers - the sensors, led controllers, motor drivers, HATs, bonnets, etc. And nearly all of these use I2C or SPI!

Wait, isn't there already something that does this - GPIO Zero?

[Yes! We like and use GPIO Zero a lot](https://gpiozero.readthedocs.io/en/stable/), its an excellent hardware interfacing library for Raspberry Pi. It's great for digital in/out, analog inputs, servos, some basic sensors, etc. In particular, one cool thing it does is thread/event management so you can have code run, say, when a button is pressed.

GPIO Zero excels at that, but doesn't cover SPI/I2C sensors or drivers, which is where we got stuck: for each sensor we had we'd write a driver in C/C++ for Arduino, CircuitPython using our hardware API, and then Python using smbus or similar.

By letting you use CircuitPython on Raspberry Pi via **adafruit\_blinka**, you can unlock all of the drivers and example code we wrote! *And* you can keep using GPIO Zero for pins, buttons and LEDs. We save time and effort so we can focus on getting code that works in one place, and you get to reuse all the code we've written already.

What about other Linux SBCs?

Plus, we're adapting and extending **adafruit\_blinka** to [support *other*boards](https://learn.adafruit.com/circuitpython-on-orangepi-linux) such as Allwinners and BeagleBone, even some smaller linux boards like Onion.io will be able to run CircuitPython code.

If you have a board you'd like to adapt [check out the adafruit\_blinka code on github](https://github.com/adafruit/Adafruit_Blinka), pull requests are welcome as there's a *ton* of different Linux boards out there! [You'll need to add a detection element](https://github.com/adafruit/Adafruit_Python_PlatformDetect) so we can tell what board you're running on, then the pin definitions into **adafruit\_blinka** above. As long as you're running a modern kernel, you'll have libgpiod for GPIO, smbus for I2C and spidev for SPI all ready to go.

Installing CircuitPython Libraries on Raspberry Pi

by [lady ada](https://learn.adafruit.com/users/adafruit2)

CircuitPython libraries and adafruit-blinka will work on any Raspberry Pi board except the compute module! That means the original 1, the Pi 2, Pi 3 or Pi Zero.

Prerequisite Pi Setup!

In this page we'll assume you've already gotten your Raspberry Pi up and running and can log into the command line

Here's the quick-start for people with some experience:

1. Download the [latest Raspbian or Raspbian Lite](https://www.raspberrypi.org/downloads/raspbian/) to your computer
2. [Burn the Raspbian image to your MicroSD card](https://learn.adafruit.com/adafruit-raspberry-pi-lesson-1-preparing-and-sd-card-for-your-raspberry-pi) using your computer
3. [Re-plug the SD card into your computer (don't use your Pi yet!) and set up your wifi connection by editing supplicant.conf](https://learn.adafruit.com/raspberry-pi-zero-creation/text-file-editing)
4. [Activate SSH support](https://learn.adafruit.com/raspberry-pi-zero-creation/text-file-editing)
5. Plug the SD card into the Pi
6. If you have an HDMI monitor we recommend connecting it so you can see that the Pi is booting OK
7. Plug in power to the Pi - you will see the green LED flicker a little. The Pi will reboot while it sets up so wait a good 10 minutes
8. [If you are running Windows on your computer, install Bonjour support so you can use **.local** names, you'll need to reboot Windows after installation](https://learn.adafruit.com/bonjour-zeroconf-networking-for-windows-and-linux#microsoft-windows)
9. [You can then **ssh** into **raspberrypi.local**](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-6-using-ssh)

[The Pi Foundation has tons of guides as well](https://projects.raspberrypi.org/en/projects/raspberry-pi-getting-started)

We really really recommend the lastest Raspbian only. If you have an older Raspbian install, run "sudo apt-get update" and "sudo apt-get upgrade" to get the latest OS!

Update Your Pi and Python

Run the standard updates:

sudo apt-get update

sudo apt-get upgrade

and

sudo pip3 install --upgrade setuptools

If above doesn't work try

sudo apt-get install python3-pip

Enable I2C and SPI

A vast number of our CircuitPython drivers use I2C and SPI for interfacing so you'll want to get those enabled.

You only have to do this *once* per Raspberry Pi but by default both interfaces are disabled!

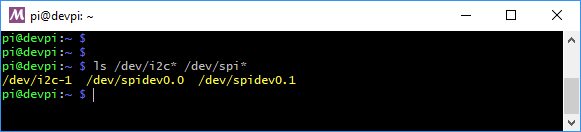
* [Enable I2C](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-i2c)
* [Enable SPI](https://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/configuring-spi)

Once you're done with both and have rebooted, verify you have the I2C and SPI devices with the command

ls /dev/i2c\* /dev/spi\*

You should see the response

/dev/i2c-1 /dev/spidev0.0 /dev/spidev0.1

[](https://learn.adafruit.com/assets/56782)

Enabling Second SPI

If you are using the main SPI port for a display or something and need another hardware SPI port, you can enable it by adding the line

dtoverlay=spi1-3cs

to the bottom of **/boot/config.txt** and rebooting. You'll then see the addition of some /dev/spidev1.x devices:

[sensors_image.png](https://learn.adafruit.com/assets/59554)

Make sure you're using Python 3!

The default python on your computer may not be python 3. Python 2 is officially discontinued and all our libraries are Python 3 only.

We'll be using python3 and pip3 in our commands, use those versions of python and pip to make sure you're using 3 and not 2

Install Python libraries

Now you're ready to install all the python support

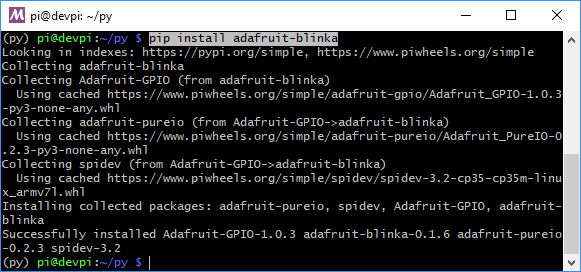
Run the following command to install the Raspberry PI GPIO library:

pip3 install RPI.GPIO

[](https://learn.adafruit.com/assets/56784)

Run the following command to install **adafruit\_blinka**

pip3 install adafruit-blinka

[](https://learn.adafruit.com/assets/56783)

The computer will install a few different libraries such as adafruit-pureio (our ioctl-only i2c library), spidev (for SPI interfacing), Adafruit-GPIO (for detecting your board) and of course adafruit-blinka

That's pretty much it! You're now ready to test.

Blinka Test

Create a new file called **blinkatest.py** with **nano** or your favorite text editor and put the following in:

 Download: [file](https://learn.adafruit.com/pages/12762/elements/2993427/download)

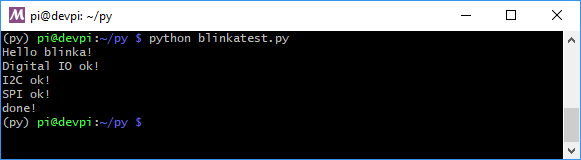
[Copy Code](https://learn.adafruit.com/circuitpython-on-raspberrypi-linux?view=all)

1. import board
2. import digitalio
3. import busio
5. print("Hello blinka!")
7. # Try to great a Digital input
8. pin = digitalio.DigitalInOut(board.D4)
9. print("Digital IO ok!")
11. # Try to create an I2C device
12. i2c = busio.I2C(board.SCL, board.SDA)
13. print("I2C ok!")
15. # Try to create an SPI device
16. spi = busio.SPI(board.SCLK, board.MOSI, board.MISO)
17. print("SPI ok!")
19. print("done!")

Save it and run at the command line with

python3 blinkatest.py

You should see the following, indicating digital i/o, I2C and SPI all worked

[](https://learn.adafruit.com/assets/56785)

Digital I/O

by [lady ada](https://learn.adafruit.com/users/adafruit2)

The first step with any new hardware is the 'hello world' of electronics - blinking an LED. This is very easy with CircuitPython and Raspberry Pi. We'll extend the example to also show how to wire up a button/switch and enable a pull-up resistor.

Even if you use a different library to create digital in/outs like GPIO Zero, there's a number of sensor libraries that use a digital pin for resetting, or for a chip-select. So it's good to have this part working!