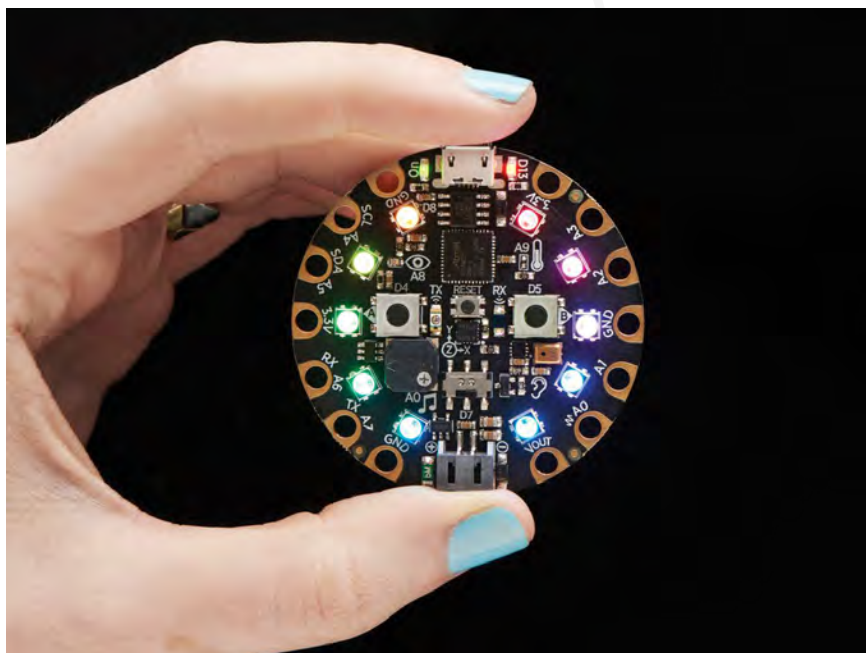


Make:

Getting Started with Adafruit Circuit Playground Express



**SPECIAL HACKADAY
SUPERCON PREVIEW!**

Mike Barela

Make:

Getting Started with Adafruit Circuit Playground Express

THE MULTIPURPOSE LEARNING
AND DEVELOPMENT BOARD
WITH BUILT-IN LEDS, SENSORS,
AND ACCELEROMETER

Mike Barela

Foreword by Limor “Ladyada” Fried

Maker Media, Inc.
San Francisco

To purchase this book in its entirety, please visit

<https://amzn.to/2CMD3vZ>

Copyright © 2018 Mike Barela. All rights reserved.

Printed in Canada.

Published by
Maker Media, Inc.
1700 Montgomery Street, Suite 240
San Francisco, CA 94111

Maker Media books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (safaribooksonline.com). For more information, contact our corporate/institutional sales department: 800-998-9938 or corporate@oreilly.com.

Publisher: Roger Stewart
Editor: Patrick Di Justo
Copy Editor: Elizabeth Welch
Proofreader: Scout Festa
Interior and Cover Designer and Compositor: Maureen Forys,
Happenstance Type-O-Rama
Indexer: Valerie Perry, Happenstance Type-O-Rama

September 2018: First Edition

Revision History for the First Edition

2018-09-15 First Release

See oreilly.com/catalog/errata.csp?isbn=978-1-68045-488-8 for release details.

Maker., Maker Shed, and Maker Faire are registered trademarks of Maker Media, Inc. The Maker Media logo is a trademark of Maker Media, Inc. *Getting Started with Adafruit Circuit Playground Express* and related trade dress are trademarks of Maker Media, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Maker Media, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps. While the publisher and the author have used good faith efforts to ensure that the information and instructions contained in this work are accurate, the publisher and the author disclaim all responsibility for errors or omissions, including without limitation responsibility for damages resulting from the use of or reliance on this work. Use of the information and instructions contained in this work is at your own risk. If any code samples or other technology this work contains or describes is subject to open source licenses or the intellectual property rights of others, it is your responsibility to ensure that your use thereof complies with such licenses and/or rights.

978-1-68045-488-8

Safari® Books Online

Safari Books Online is an on-demand digital library that delivers expert content in both book and video form from the world's leading authors in technology and business. Technology professionals, software developers, web designers, and business and creative professionals use Safari Books Online as their primary resource for research, problem solving, learning, and certification training. Safari Books Online offers a range of plans and pricing for enterprise, government, education, and individuals. Members have access to thousands of books, training videos, and prepublication manuscripts in one fully searchable database from publishers like O'Reilly Media, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, Course Technology, and hundreds more. For more information about Safari Books Online, please visit us online.

How to Contact Us

Please address comments and questions to the publisher:

Maker Media, Inc.
1700 Montgomery Street, Suite 240
San Francisco, CA 94111

You can send comments and questions to us by email at books@makermedia.com.

Maker Media unites, inspires, informs, and entertains a growing community of resourceful people who undertake amazing projects in their backyards, basements, and garages. Maker Media celebrates your right to tweak, hack, and bend any Technology to your will. The Maker Media audience continues to be a growing culture and community that believes in bettering ourselves, our environment, our educational system—our entire world. This is much more than an audience, it's a worldwide movement that Maker Media is leading. We call it the Maker Movement.

To learn more about Make: visit us at make.co.

Contents

Foreword	ix
Preface	xi
1 Introducing Circuit Playground Express	1
2 A Tour of Circuit Playground Express	7
Circuit Playground Express Outputs	10
Circuit Playground Express Inputs	12
Powering Your Circuit Playground Express	13
Operating System Software Setup	17
Chapter Questions	20
3 Getting Started with Microsoft MakeCode	21
Connecting a Circuit Playground Express to a Computer	22
MakeCode: Your First Program	24
Uploading MakeCode to Circuit Playground Express	30
Modifying a Program	39
Saving a Program	44
Under the Hood: JavaScript	46
Wrap-Up	48
Chapter Questions	48
4 Microsoft MakeCode and Interactivity	49
Using Buttons	50
Shake, Rattle, and Roll	63
Making the Accelerometer Display Multiple Animations	67

Using the Slide Switch	70
Your Turn: Slide Switch	74
JavaScript	74
Wrap-Up	76
Chapter Questions	76
5 Advanced Microsoft MakeCode	77
Sound and Music	78
Listening for Sounds	85
Wrap-Up	92
Chapter Questions	92
6 Coding with CircuitPython	93
CircuitPython vs. Other Python Implementations	95
Installing CircuitPython	96
Text Editors	102
Eject or Sync the Drive after Writing	103
Using Mu	105
Creating Python Code	107
Examining the CircuitPython Blink Code	114
Output from Circuit Playground Express to the Computer	115
The Adafruit Circuit Playground Express Library	119
Running Code on Express via the REPL	124
Wrap-Up	126
Chapter Questions	126
7 Using the Circuit Playground Express CircuitPython Library	127
Reading Sensors	128
File Input and Output	131
Capacitive Touch and Music	142
Emulating a Computer USB Keyboard	155

Mouse Emulation	161
Wrap-Up	165
Chapter Questions	165
8 Using the Arduino Development Environment	167
The Arduino Programming Language	170
Installing the Arduino IDE	171
Structure of an Arduino Program	177
Uploading Code to Circuit Playground Express	184
The Circuit Playground Arduino Library	191
Circuit Playground Library Functions	195
Example Code	199
Libraries and Compatibility	201
Wrap-Up	204
Chapter Questions	205
A Troubleshooting	207
USB Cable and Power Issues	207
Connectivity Issues	210
CircuitPython Issues	213
Arduino IDE Issues	215
Common Arduino Library Problems	215
Error Messages	218
Usage Issues	219
Manufacturer Support	221
B Reference Materials	223
On the Internet	223
Publications	225
About the Author	227
Index	229

Foreword

The story of Circuit Playground begins maybe eight years ago. Adafruit was still an apartment company then. My partner and I were chatting with a middle school superintendent who told us that the school was being pitched STEM (science, technology, engineering, and mathematics) education products for its students (the products were similar to tablets with sensors that could plug in). But at \$500 each, the school could afford only one per classroom. So twenty-plus kids would have to share.

At the time, Arduino was becoming popular—it's a lot less expensive! But younger students struggled with learning C++ (especially if they were coming from block-based Scratch programming), and the setup could get complicated since Arduino requires a special development environment.

For a long time, I didn't have a solution to these problems. The slick technology was just too expensive, and the low-cost educational kits were too hard to use. But eventually, enough stuff was invented (low-cost ARM Cortex microcontrollers! NeoPixels! Embedded Python!) that we were able to make the ultimate circuit board for teaching coding and electronics.

That's where Circuit Playground comes in. Easy to use, fun to program, and affordable for any student, it works with the Mac, Windows, Linux, Chrome OS, and even Android! You can use it at home, at school, at work, or on a library computer—no software needs to be installed.

We poured all the know-how and experience we've had over 10 years of selling educational electronics to create something

for everyone. Whether you want to build cosplay props, scientific experiments, robotics, or spy gadgets—in drag-n-drop Microsoft MakeCode, interpreted CircuitPython, or Arduino—Circuit Playground Express will be your companion as you learn and create.

—Limor “Ladyada” Fried, founder and engineer, Adafruit

SPECIAL
HACKADAY
SUPERCON
PREVIEW

Preface

Adafruit Circuit Playground Express provides a low-cost way to explore programming, sensing, and interaction. The Express is a microcontroller-based electronics and software development board. It is programmable in Microsoft MakeCode, JavaScript, and Python and with the Arduino development environment. Its built-in motion, temperature, and light sensors let Circuit Playground Express sense the world around it. Its 10 NeoPixel lights and speaker allow Circuit Playground Express to communicate with the outside world.

Circuit Playground Express is different from many beginning electronics available today. Out of the package, Circuit Playground Express can be connected to a computer that runs any operating system. Load Microsoft MakeCode in an Internet-connected web browser, and in less than 15 minutes you'll have an interactive project all your own.

Think—would you fancy clothes or shoes with LEDs that dance to movement and music? Would you like a musical synthesizer that plays your choice of sounds, even using fruit as your input? Perhaps a light-up pin that makes Star Trek–like sounds when tapped? All these and many, many more can be built using Circuit Playground Express right out of the package!

This book provides the information to get you started using Circuit Playground Express quickly. The information and ideas in the book may be the foundations for your own projects and explorations.

WHO THIS BOOK IS FOR

This book is for the enthusiast, the student, the curious person who wishes to expand their knowledge of making through interactivity, sensing, lights, or sound.

Skills that are useful in working through this book:

- ✱ A knowledge of the fundamentals of what software and hardware are.
- ✱ Experience with desktop or laptop computers running an operating system such as Microsoft Windows, Apple macOS, Chrome OS/Chromebook, or Linux. Skills include navigating a filesystem and selecting specific files to use.
- ✱ Use of a graphical Internet web browser. Many are available, including Chrome, Firefox, Safari, Internet Explorer, and Microsoft Edge.
- ✱ Use of a text-based editor on one of the listed operating systems and the ability to open a text file, change the file, and save the file both to the computer disk and to a flash drive connected to the computer.

Working with Circuit Playground Express is suitable for beginners who do not know electronics or programming. After you finish reading, you can use this book as a reference for the techniques presented.

PREPARATION

There is no required reading to work with this book, but here are some suggested resources that you may draw on to better understand particular subjects as the book progresses.

MakeCode

The Microsoft MakeCode.org website (<https://makecode.com/#learn>) is a good reference. Adafruit has a free tutorial on learning Makecode (<https://learn.adafruit.com/makecode>). Adafruit continually publishes new projects and tutorials on Circuit Playground Express at learn.adafruit.com (<https://learn.adafruit.com/>). Finally, Adafruit has support forums for assistance at forums.adafruit.com (<https://forums.adafruit.com/>).

Python Basics

The website python.org (www.python.org) provides free materials (www.python.org/about/gettingstarted/) to help you learn the Python programming language.

Arduino

The book *Getting Started with Arduino, Second Edition*, by Massimo Banzi (co-creator of Arduino), is a good resource to start with. I also recommend the Adafruit Learn Arduino series (<https://learn.adafruit.com/lesson-0-getting-started>), available for free online. Both offer an introduction to the Arduino open source electronics prototyping platform, including programming.

WHAT YOU WILL WANT TO HAVE ON HAND

To program Circuit Playground Express, you will need a Windows, Mac, or Chromebook computer with a USB port. You need Internet access to run the Microsoft MakeCode editor and to download example code, rather than typing it in yourself.

A Good USB Type A Male-to-Male Micro-B Cable

I cannot stress this enough: get a *good* USB cable for programming Circuit Playground Express. Please consider buying a substantial USB type A male end to type Micro-B male cable, 3 feet (1 meter) long or so (longer or shorter is fine). Frustration and questions come when unworkable USB cables are pressed into service. Such cables, more often than not, do not have the USB data wires required for communicating between the computer and the Circuit Playground Express. Worn cables may work intermittently when bent just right—never good. A good cable will save you hours of grief.

Overall, working with Circuit Playground Express requires very little knowledge other than how to observe and how to innovate.

CONVENTIONS USED IN THIS BOOK

The following typographical conventions are used in this book:

- ✱ Menu selections are shown by a series of options separated by the → symbol (e.g., choose Tools → Board).
- ✱ Keyboard entries are shown in boldface (for example, enter **adafruit**).
- ✱ Monospaced font is used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, filenames, file extensions, databases, data types, environment variables, statements, and keywords.

Examples

This book is here to help you get your job done. In general, you may use the code in this book in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing a CD-ROM of examples from *Make: books* does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example: *Getting Started with Adafruit Circuit Playground Express*, by Mike Barela (Maker Media). Copyright 2018, 978-1-68045-485-7.

If you feel your use of code examples falls outside fair use or the permission given here, feel free to contact us at bookpermissions@makermedia.com.

6

.....

Coding with CircuitPython

In this chapter, an alternate way to program Circuit Playground Express is demonstrated using a language called *Python*. Python is the fastest-growing programming language in use today and is taught in schools and universities. It's a high-level programming language, which means it's designed to be easy to read, write, and maintain. Figure 6-1 shows Blinka, the CircuitPython mascot.



FIGURE 6-1. The Adafruit CircuitPython mascot: Blinka

The change from using the graphical Microsoft MakeCode to using text-based programming may seem like a huge one. It is not. The Python programming language has been gaining a steady following, significantly in the Raspberry Pi community. Why? Because Python is easy to use, and because the language was designed from the start to be easy to improve, extend, and grow. The same Python code can run on a small wearable, on to other computers, on up to a supercomputer, with perhaps only a slight change.

Python includes modern programming commands and it supports code extensions, called *modules*. (In some computer languages, like Arduino, these are called *libraries*.) Modules are code packages that can be used by a Python program to perform specific tasks. For example, there are modules to perform complex number crunching or to graphically plot data. And nearly all modules are open source software—code available on the Internet at no cost and freely shared.

Module vs. Library

There's some discussion in the computing field as to whether code that is imported into Python is a module or a library. To remain consistent with CircuitPython documentation, I'll use the term *library* most often. Just think of the terms as interchangeable for this book.

Unlike the Arduino environment, where all coding is done on a desktop or laptop, compiled into machine code, and then loaded onto the circuit board, Python is an *interpreted language*. This means that the hardware can interpret and act on each command you type, practically instantaneously. There's no need to compile and upload your code to see if it works.

CircuitPython provides interactivity via a Read–Eval–Print Loop (REPL, pronounced *rep-ul*). On your computer you can type Python commands into the REPL, and the board will process and respond to each line of programming entered. This allows the user to see what specific commands do in real time rather than performing multiple steps to get code into a processor for execution.

CIRCUITPYTHON VS. OTHER PYTHON IMPLEMENTATIONS

CircuitPython is the implementation of Python created by Adafruit for several products, including Circuit Playground Express. It is a *fork* (derivative) of MicroPython, a version of Python written by Damien George to run on microcontrollers.

CircuitPython adds hardware support for a range of Adafruit Industries microcontrollers. It allows users with limited hardware experience to easily program their devices. No previous experience necessary—it's really simple to get started!

All CircuitPython code is run from the Circuit Playground Express internal flash drive/thumb drive, the space used previously to put MakeCode onto the board.

CircuitPython excels at the following:

- ✱ **Very fast development:** Write the code, save the file, and it runs immediately. No compiling required.
- ✱ **REPL:** You can start interactive programming with the REPL.
- ✱ **Easy code changes:** Since your code lives on the flash drive, you can edit it whenever you like, and you can also keep multiple files around for easy experimentation.
- ✱ **It's Python!** CircuitPython is completely compatible with Python (it just adds hardware support).

- ✱ **Strong hardware support:** There are many more modules for external sensors and capabilities than in MakeCode (but not as many as Arduino, yet).
- ✱ **File storage:** CircuitPython's data storage ability makes it great for data logging, playing audio clips, and otherwise interacting with files. MakeCode doesn't currently have support for file storage, and Arduino has limited support at present.

See Appendix B for more Python resources.

INSTALLING CIRCUITPYTHON

If you are sure your board already has the latest CircuitPython release, you can skip this section (for example, a teacher says you are all set or you have already placed CircuitPython on your Circuit Playground Express).

If you would like the latest version of CircuitPython, go ahead and follow along. Updating the software is the same as installing a fresh copy.

How do you know which version of CircuitPython is on your Circuit Playground Express? Connect your Circuit Playground Express to your computer. After a moment, the flash drive CIRCUITPY should appear (Figure 6-2). If your Express does not show a new CIRCUITPY drive but shows a CPLAYBOOT drive, then it needs to have CircuitPython loaded; see the how-to in the next section.

If the board is providing a CIRCUITPY drive, you should see a file on the drive called `boot_out.txt`; see Figure 6-3.

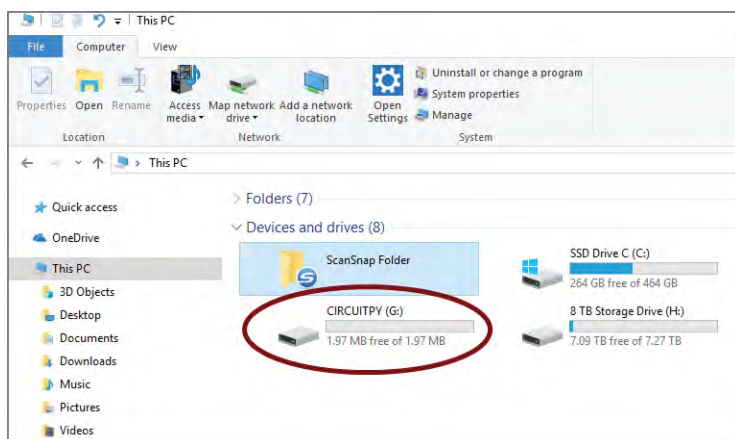


FIGURE 6-2. The CIRCUITPY flash drive in Windows Explorer

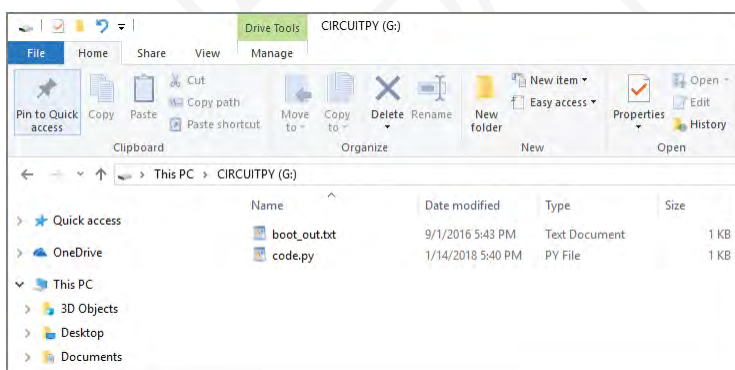


FIGURE 6-3. The `boot_out.txt` and `code.py` files on the CIRCUITPY flash drive

If you open the `boot_out.txt` file (usually by double-clicking it with the mouse), the file contains information on the version of CircuitPython placed on the board. When I updated my board for this book, the `boot_out.txt` file contained the following:

```
Adafruit CircuitPython 3.0.0 on 2018-05-04; Adafruit Circuit Playground Express with samd21g18
```

You can compare the version number (in this case 3.0) with the latest version on the Adafruit website and decide if a more up-to-date version is available.

In the off chance you believe the board has failed and CircuitPython no longer works as it did, you can install the latest version to set it up fresh.

Downloading the Latest Version of CircuitPython

When you looked for CircuitPython on your Circuit Playground Express, you might not have found it. If there is no CIRCUITPY drive, there might be a CPLAYBOOT drive. This indicates the board was being used for MakeCode or something else (see Figure 6-4). No worries—this is easily fixed. You can also check to ensure your CircuitPython installation is up to date.

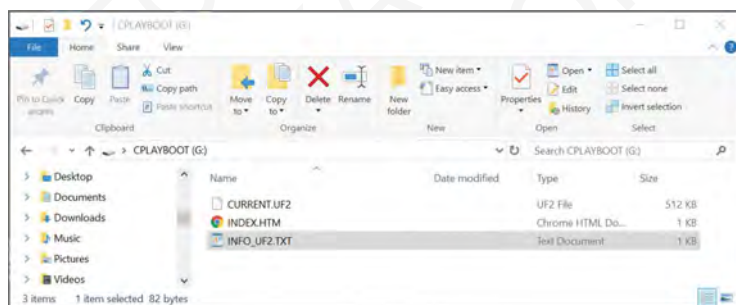


FIGURE 6-4. When you plug Circuit Playground Express in and you get a CPLAYBOOT drive rather than a CIRCUITPY drive, don't worry—that's easily fixed.

Use your Internet web browser to go to <https://github.com/adafruit/circuitpython/releases/latest>. Scroll down to the list of CircuitPython files, and choose the file that contains the text `circuitplayground_express` in the filename. When you click the file, the operating system will display a box that asks you where to save the UF2 file that is the CircuitPython code. You can save

it to any file directory you like; just remember where you save it. On Windows this can be the desktop or the Downloads or Documents folder. You can also save the file to a personal flash drive.

Remembering the Gotchas from Chapter 2

If you are running Windows 7, you will need a software driver installed to have your computer recognize the Circuit Playground Express board correctly.

Be sure you use a high-quality USB-to-MicroUSB cable with both power and data lines. Old cables, or cables that are used only to charge another device, will not work and will almost certainly lead you to frustration.

Plug your Circuit Playground Express into your computer and ensure the green power LED is on. Find the Reset button on your board. It's the small button located in the center of the board.

Tap this button once to enter the bootloader. The NeoPixels on the board will flash red and then stay green. A new drive will show up on your computer. The drive will be called CPLAYBOOT (see Figure 6-5).

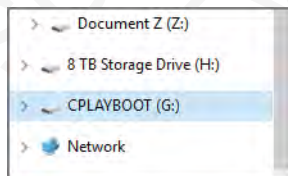


FIGURE 6-5. The CPLAYBOOT drive appears when you reset the board.

If you do not see this drive on your computer, don't be discouraged. Tap the Reset button twice. The rhythm of the taps on the Reset button needs to be correct, and sometimes it takes a try or

two. If you have a Circuit Playground Express and it's fresh out of the bag, pressing the button once will probably do it.

The number of presses to get the green circle and CPLAYBOOT drive depends on what software was used previously: MakeCode, CircuitPython, or Arduino. There is no “wrong” way—use one press or two, as long as the CPLAYBOOT drive shows up in the operating system.

Take the UF2 file you saved from the web and drag it onto the CPLAYBOOT drive (or otherwise initiate a copy of the file to the Circuit Playground Express CPLAYBOOT drive). The red D13 LED will flash as the file is transferred and the NeoPixels will blink and then go out.

If everything is successful, your computer will show a new flash drive named CIRCUITPY. This is your indication that you now have CircuitPython ready on your Circuit Playground Express. Your computer, for instance Windows 10, may pop up a message that it is setting up a new device called Circuit Playground. This is fine.

On the drive, you should see a file named `boot_out.txt` on the CIRCUITPY drive.

Now it's time to check whether you have the latest version of CircuitPython on your Circuit Playground Express. Open the `boot_out.txt` file, and you will see information on the version of CircuitPython that was placed on the board:

```
Adafruit CircuitPython 3.0.0 on 2018-05-04; Adafruit  
CircuitPlayground Express with samd21g18
```

It is possible that your board will show a version number greater than 3.0. Version 3.0.0 was released in 2018 (with additional releases planned after that) as Adafruit provides additional functionality. `samd21g18` looks like a gamer tag but it is the manufacturer's name for the microcontroller on the board.

Do I Have to Install CircuitPython Regularly? Must I Upgrade?

You have to install CircuitPython only once unless you are switching back and forth between CircuitPython, MakeCode, and Arduino. Installing once, you are free to code all you like without going through the install process again until you want to upgrade. Upgrades from Adafruit may come out periodically, adding additional features or fixing issues noted by users. Often, it is worth upgrading if you want to use new features or there is a major update. If you have a project you intend to use and it works well, consider keeping things the way they are.

I'm Having Weird Installation Issues!

Don't worry! The problem could be a corrupt flash drive, which is not the end of the world (or your board!). If this happens, follow the steps found in Appendix A.

Now that the Circuit Playground Express board is ready, it's time to code in Python. First, though, you need to make a decision. The process of creating code files involves typing in text, much like the JavaScript we saw in MakeCode. What software should you use to create the code files? Python commands (code) will be saved into a text-based file. For that we need a text-editing program.

TEXT EDITORS

Unless you are typing in the REPL (more on using the REPL later in the chapter), you'll want to type your CircuitPython code in a *text editor*. A text editor is a program on a PC, Mac, Linux, or Chromebook that accepts text input and allows you to edit the text and save the final result to disk.

No one wants to type complex programs into the REPL over and over. One typing mistake and we'd have to start all over. If we store the program text in a file, we can save it for later use.

Text editors come standard with all operating systems:

- ✱ PC/Windows: Notepad, Wordpad, Microsoft Word
- ✱ Linux: Nano, Vim, EMACS
- ✱ Mac: TextEdit, Pages, TextMate
- ✱ Chromebook: Caret, Google Docs, Writebox, Text

If you are in a learning environment, your teacher will often tell you which editor to use and guide you on how to use it. If you're going through on your own, you are free to select your own text editor. Full-fledged word processors such as Microsoft Word, Google Docs, and Mac Pages do not save plain text by default. A more basic editing program often works best.

Try the Mu Editor (if possible). Mu is a simple editor that runs on Windows, macOS, Raspberry Pi, and Linux (the list may expand to other platforms as the developers have time). A serial console is built right into the Mu program so that you get immediate feedback from your board's serial output and easily use the REPL in the same program!

If you find you cannot use Mu, use the text editor of your choice.

For Chromebook, the examples will be using the Caret editor. A separate terminal emulation program is needed to type commands into the REPL and to receive REPL and program output

from CircuitPython. This book will use the Chromebook terminal emulator Beagle Term to perform serial input and output. Both applications are free in the Chrome OS app store.

NOTE Mu will be shown in most examples. Mu is the recommended editor for Windows, Mac, and Linux. Please consider using it (unless you have a favorite editor already!). This eliminates the need for using two programs, a text editor and a terminal emulation program, to interact with the Circuit Playground Express serial input and output. Mu is not required. If you are more experienced, another editor and a terminal program will work fine.

WARNING Ensure that you use an editor that writes out files completely when you save a file to disk. It is so easy to write your code in an editor and fail to save the code to disk. Many “good” editors will prompt you to save your work, but some do not. Both Notepad (the default Windows editor) and Notepad++ can be slow to write. You need to click the save icon to ensure your data is saved, and you must be sure to eject the drive.

EJECT OR SYNC THE DRIVE AFTER WRITING

If you are using a problematic text-editing program, not all is lost! You can still make it work.

On Windows, you can eject or safely remove the CIRCUITPY drive by right-clicking the drive in File Explorer and clicking Eject. It won’t actually eject, but it will force the operating system to

save your file to disk. On Linux, use the `sync` command in a terminal to force the write to disk.

When a program edit is complete, save a copy somewhere safe, such as a hard drive or data flash drive, to ensure that you have a copy for later use.

Installing Mu on Windows or Mac

The first step is to download the latest version of Mu. If you are using Windows, you must be running Windows 7 or higher. For macOS you must be running 10.12 (Sierra) or higher (Mac users with lower versions can try the Linux instructions that follow, but that is not guaranteed to work, according to the author of Mu).

The main Mu repository is located on the web at <http://codewith.mu>.

Select the latest version for your operating system. Download and save the Mu installation file to your desktop, download folder, or wherever is handy. Run the installation program. The installation process is operating system dependent (Figure 6-6). Windows installation programs ending in `.exe` or `.msi` can be run by double-clicking. macOS has its own install package.



FIGURE 6-6. The Mu program icon on Windows (left) and Mac (right) when placed on the desktop. Icons are subject to change by the Mu developers.

Once you have the program installed on your computer, you're ready to start coding Python.

Installing Mu on Linux

Each Linux distro is a little different, so use the following as a guideline. See https://codewith.mu/en/howto/install_with_python for details.

1. Open a terminal window.
2. Mu requires Python version 3. If you haven't installed Python yet, do so via your command line using something like `sudo apt-get install python3`.
3. You'll also need pip3 (or pip if you have only Python 3 installed); try running `pip3 --version`. If your system does not have pip installed, run `sudo apt-get install python3-pip`.
4. Finally, run `pip3 install mu_editor`.
You can now run Mu directly from the command line.

USING MU

Mu attempts to automatically detect a Circuit Playground Express plugged into a computer. Before starting Mu, please plug in your Circuit Playground Express and make sure it shows up as a CIRCUITPYTHON drive in your computer's file explorer.

Once Mu is started, you will be prompted to select your mode (Figure 6-7). Please select Adafruit CircuitPython.

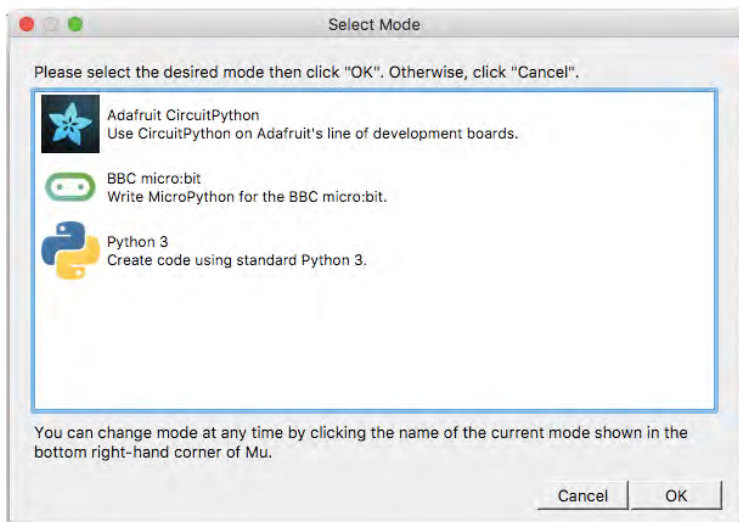


FIGURE 6-7. The Mu Select Mode screen

If you get a warning that the Circuit Playground Express cannot be found (Figure 6-8), ensure you have the board plugged into your computer. Check to see if the CIRCUITPY drive shows up in the available disk drives on your computer. If you are still having issues, see Appendix A for troubleshooting tips.



FIGURE 6-8. The warning that's shown if your Circuit Playground Express is not plugged in when Mu is started

You should now see the main Mu screen (Figure 6-9).

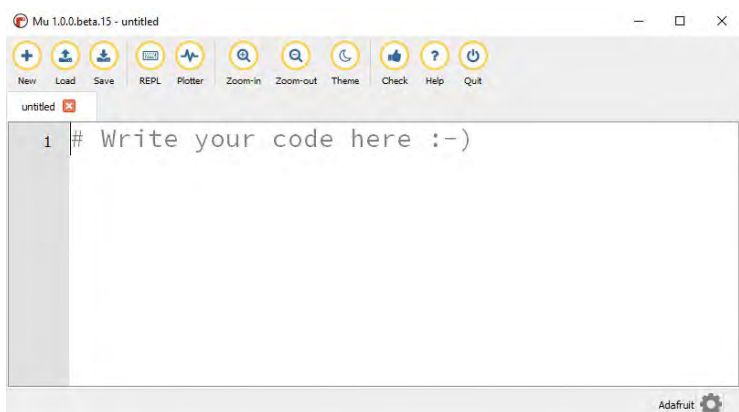


FIGURE 6-9. The Mu main editing window

Now that Mu is available, it is time to start coding Python.

CREATING PYTHON CODE

Plug your Circuit Playground Express via a known good USB cable into your computer. Your file explorer should show that a new flash drive is available named CIRCUITPY. (If you do not see the drive, see the earlier section “Downloading the Latest Version of CircuitPython” or Appendix A.)

Start the Mu editor. Usually you double-click the Mu icon in Windows or click the Mac icon. If you are on a Chromebook, run Caret or the editor you have chosen.

As with MakeCode, a good first program to create is one that makes a light on the board blink. Here are the general steps:

1. You set up the board environment before starting the main forever loop.
2. You create a loop that continually runs—in this case, turning an LED on, waiting, turning the LED off, waiting, then rerunning the loop.

Generic Python code does not know it will be running on a Circuit Playground Express. To assist Python with hardware-specific tasks, the creators of Circuit Playground Express have easy-to-use code that performs operations on the board, such as performing digital input and output.

In the larger Python world, thousands of modules and libraries are available for performing a wide variety of useful tasks. For the smaller CircuitPython, memory size limits the number of libraries a bit. Adafruit is committed to providing a wide variety of functionality for their CircuitPython products.

Why use a library? Libraries are great for two reasons:

- * Libraries allows code portability. If you take your CircuitPython code for Circuit Playground Express and place it on another CircuitPython product such as an Adafruit Feather M0, the code will run (if you haven't used board-specific capabilities). The underlying hardware may be very different, but the authors of the library have done the translation from Python to hardware coding.
- * Libraries allow a person to focus on their project and not the nuts and bolts of a specific hardware architecture. It is but one premise of open source software: someone has taken the time to write (hopefully useful) code to do something you would also like to do, such as code that makes a motor turn on and off. Coding the low-level motor control via hardware can be rather difficult. If the project designer can import a library to easily code something like `motor.on` and `motor.off`, the designer can focus more on a project and not have to intimately know how to control the motors on the circuit level.

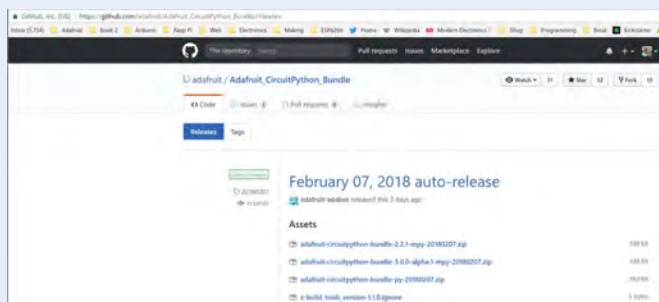
The programmers at Adafruit have written a library for CircuitPython to interact with the Circuit Playground Express

board's built-in hardware. It is called `adafruit_circuitplayground`.
.express. The library has a number of useful functions, which we
will use in examples in this chapter.

Installing the Adafruit CircuitPython Libraries

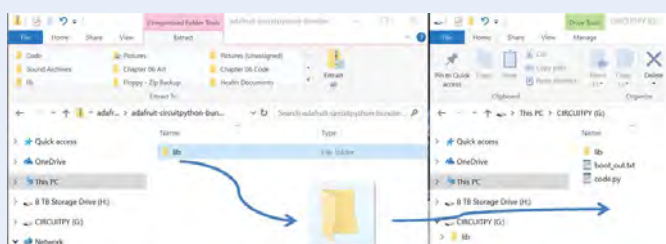
Adafruit and contributors have created a large number of libraries to handle the built-in hardware on Circuit Playground Express and other devices that may be added onto the board (like a display, for example). You can find the file necessary to add this capability here: https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases.

You will need to determine the CircuitPython release you are running on your board, which you can do by opening the `boot_out.txt` file on the CIRCUITPY flash drive when you plug the board in. As of this writing, there were versions for a 3.0.0 branch and a 3.0.0 branch. Since I am running the 3.0.0 version of CircuitPython, I chose the corresponding 3.0.0 zip file and saved it on my computer.



Open the zip file and you will see many small files used for a variety of input and output devices.

With the zip file open on your computer, highlight and copy the **lib** directory (which contains all the files and subfolders we need) to the CIRCUITPY drive. It will take some time and consume about 400KB of space. This still leaves plenty of room on the drive for programs. Be sure to use the **eject** function of your file explorer program (**sync** in Linux) to ensure all writes are made to the Circuit Playground Express flash memory before disconnecting the board from USB.



Now Circuit Playground Express should be set to use a wide variety of prebuilt libraries. If you ever need to refresh the library with the latest code, follow the same procedure.

To use the entire code library, you include the following command at the top of a Python program:

```
import adafruit_circuitplayground.express
```

Whenever you want to use a function in the module, you will need to type the function name, such as when reading the A Button:

```
button_read = adafruit_circuitplayground.express.button_a
```

Fortunately, we can shorten the function calls due to some Python, behind the scenes. Here is how to refer to the

`adafruit_circuitplayground.express` module by using the handy acronym `cpx`:

```
from adafruit_circuitplayground.express import cpx
# Objects to Circuit Playground Express objects can now be
# referred to their abbreviated form
button_read = cpx.button_a
```

Much better to read—and less typing.

For the blinking D13 LED, we'll use the Circuit Playground Express library in CircuitPython.

Type the following code into your text editor:

```
import time
from adafruit_circuitplayground.express import cpx
while True:
    cpx.red_led = True
    time.sleep(1)
    cpx.red_led = False
    time.sleep(1)
```

For the lines after `while True:`, you should *indent* the code. Indenting (putting space before the text on a line) is done by either typing four spaces or pressing the Tab key. This informs Python that the code should all be contained in the statement above it. The `while True:` provides a forever-style loop as in Make-Code or the `loop()` function in Arduino. The value `True` will always be true so `while True:` will always loop all the statements within it.

To Tab or to Space

Some coders prefer to indent code using spaces. Many Python programmers indent with the Tab key, which is equal to a set number of spaces. Neither is wrong, but all agree that you should not mix tabs and spaces or you may end up with a Python error that will not be obvious to fix. Some text editing programs might put an actual tab character into the text file. The editor might convert a tab to multiple spaces. Consistency is the important thing. Using Tab will almost never make someone later editing your code question your choice.

Now save your program to a disk. You will want a place to store a copy of your code for later use; this can be a hard disk, a flash drive, or online storage like Google Drive.

When you save the program, use the filename `blink.py`. The `.py` at the end is called the file extension, and it lets you and others know the text file contains Python code. Also, save the program as `code.py`. This is the name CircuitPython recognizes as the current program you wish to run.

Program Names

The CircuitPython file you copy over to run on Circuit Playground Express should always be called `code.py`. This is so the Python interpreter knows the name of the code you want to run.

If you use `mycoolprog.py` or anything else, the board will not know that file is the code you wish to run. Feel free to use a more descriptive name on your backup storage device copies—for example, `music-on-tilt.py`.

The CircuitPython authors state there are four filename options for the code the board will run: `code.txt`, `code.py`, `main.txt`, and `main.py`. CircuitPython looks for those files, in that order, and then runs the first one it finds. The author and Adafruit suggest using `code.py` as your code file.

Still, it is important to know that the other options exist. If your program doesn't seem to be updating as you work, make sure you haven't created another code file that's being read instead of the one you're working on.

Throughout the book, we will use `code.py` as the CircuitPython program to run.

Running the Python Code

Okay, so at this point you have your Python program. Let's get that code running.

Plug in your Circuit Playground Express, via USB, to your computer. The board should show up as a flash memory drive named CIRCUITPY.

Copy the file `code.py` you saved earlier via your computer's file program over to your CIRCUITPY drive.

If you are using the Mu editor, you can use the save button to save the file to Circuit Playground Express if you named the program `code.py`. If you need a "Save as" function to make the copy, double-click the filename on the Mu tab that contains the program name. A dialog box will appear allowing you to name and save the code wherever you wish.

The red D13 LED next to the USB connector should now be flashing. The Python code is working!

On the off chance your code is not working:

- * Be sure the green power LED is on and you see the CIRCUITPY flash drive.
- * Be sure your file is called `code.py` (and not `blink.py` or something else). Unlike MakeCode filenames, the `py` file on the board must be named `code.py`.
- * You can press the Reset button and that should restart the Python code if it is not running already.
- * Double-check your Python program against the earlier listing. The indented text should not mix tabs and spaces. You must indent the text as shown. In Python, that indicates the text is in a loop like MakeCode.

EXAMINING THE CIRCUITPYTHON BLINK CODE

Without going into the details of the entire Python programming language, we can understand what the code is doing by comparing the code to similar actions in MakeCode.

Modules are similar to the code blocks in MakeCode that allow you to select blocks of a specific type—for example, `LOOPS` for blocks that perform looping actions, `LIGHT` for blocks that turn LEDs on and off, and so on. What we do not have at hand are the range of statements that can be created in each library—the library building blocks. Let's explore them.

The `while True:` statement is the same as the `forever` loop in MakeCode. The code within the `while True:` loop will be executed “forever.” The code that follows `while` can be any mathematical statement that evaluates to a `True` or `False` condition. It could be `x < 3` or another comparison that is evaluated as `True` or `False`. Here we just use `True`, so in essence `while True:` always loops (it never exits because the constant value `True` is never `False`).

The code in the `while` loop is similar to the code used in MakeCode. The LED is turned on by setting the value of the object `cpx.red_led` to `True`. The `time` function waits 1 second. The LED is turned off by setting the value of the object `cpx.red_led` to `False`. Then another second elapses before starting the loop again.

With additional examples, you'll see more Python statements and library functions that build on the work of the blink example.

Using the Internal (Frozen) CPX Library

The CPX library is always available to CircuitPython programs. As of CircuitPython version 2.3.0, CPX is a *frozen library*. A frozen library is part of the core of CircuitPython as of version 3.0.0 and higher. No file in the Circuit Playground Express `/lib` directory is required to support the CPX library.

If you have upgraded from a CircuitPython version prior to 3.0.0 and you have library files in the `/lib` sub-folder on the CIRCUITPY drive, you will want to delete the CPX library in `/lib`. You do this by deleting the directory and files in `/lib/adafruit_circuitplayground/`. That way, you can be sure that you are using the frozen version of the CPX library.

OUTPUT FROM CIRCUIT PLAYGROUND EXPRESS TO THE COMPUTER

Earlier when we programmed Circuit Playground Express using MakeCode, there were no blocks that allowed the user to interactively communicate with the board. The computer allowed you to code and let you see what that code would do. But there was no ability for the code to take computer input or format output to the computer. Fortunately, CircuitPython has greater flexibility when performing input and output.

Up until now, the USB connection has provided two functions:

- * Power the board via the USB cable
- * Enable loading code and reviewing files on the device as if the board is a flash memory drive

Universal Serial Bus (USB) provides a number of other functions that are very useful. This capability can be used by

programming in CircuitPython (or later in Arduino). Two additional functions that are extremely useful are

- ✱ Input and output to the connected computer over USB—typically called serial communications
- ✱ Human Interface Device (HID) mode, which allows the board to emulate devices such as a keyboard or a mouse

Serial Output

Time to modify our first Python program. Add the line that starts with `print` below the `while True:` statement to your code:

```
import time
from adafruit_circuitplayground.express import cpx
while True:
    print("Hello CircuitPython!")
    cpx.red_led = True
    time.sleep(1)
    cpx.red_led = False
    time.sleep(1)
```

Please make sure it is indented and typed correctly with two double quotation marks and two parentheses.

The `print` statement will send the function argument to the serial output (out the USB port back to your computer).

Here is where using the Mu editor helps. If you do not have Mu, skip ahead for how to get the serial output.

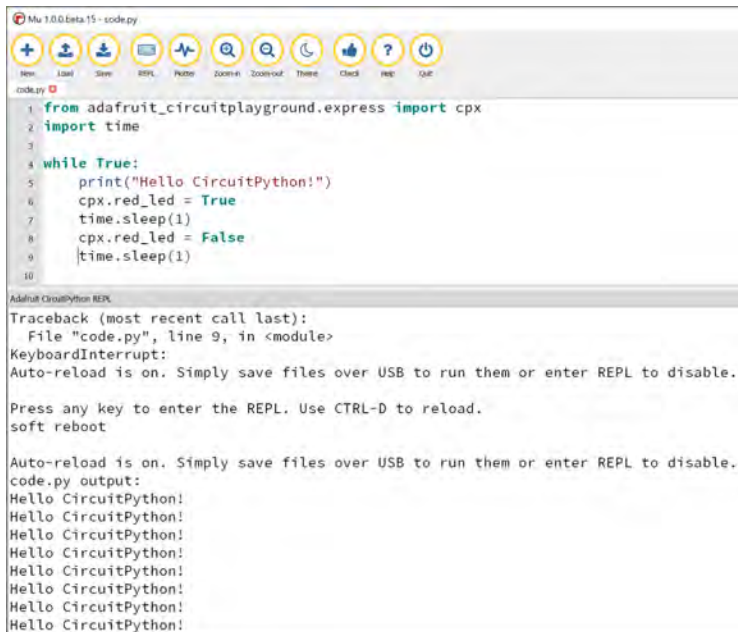
Click the Serial icon, the fifth icon from the left with a double arrow icon on it (see Figure 6-10).



FIGURE 6-10. The Serial icon in the Mu editor, fifth from the left, has a double arrow icon.

The Serial screen acts both as a command input/output window and as an interactive REPL Python environment.

Once you have typed in the print line, your screen should look similar to the one in Figure 6-11. You can adjust the size of each window; in Figure 6-11, I expanded the Serial window to show more of what is being output by Circuit Playground Express. Left-click and hold the mouse on the light gray line between the code and Serial windows, adjust, and then release the left mouse button.



The screenshot shows the Adafruit CircuitPython IDE interface. At the top, there's a toolbar with icons for New, Load, Save, REPL, Paste, Zoom In, Zoom Out, Theme, Check, Help, and Quit. Below the toolbar, the code editor displays a Python script named `code.py` with the following content:

```
1 from adafruit_circuitplayground.express import cpx
2 import time
3
4 while True:
5     print("Hello CircuitPython!")
6     cpx.red_led = True
7     time.sleep(1)
8     cpx.red_led = False
9     time.sleep(1)
10
```

Below the code editor, the Serial window is open, showing the output of the program. It displays a traceback message indicating a `KeyboardInterrupt` at line 9, followed by instructions about auto-reload and the REPL. The serial output shows the program printing "Hello CircuitPython!" repeatedly.

FIGURE 6-11. With the Serial window open, serial output can be seen.

Every two seconds, you should see the words *Hello, CircuitPython!* display on the screen. If you don't see the text at the bottom of the screen, press `Ctrl+D` to restart the program.

The `print` function allows the user to print text and numbers through the USB port on Circuit Playground Express to the host computer. With the Mu editor, the text shows up in the Serial window.

If you are not using Mu, the text can be seen by any program that opens Circuit Playground Express as a serial device. On a Mac, the screen command opens up a compatible serial connection. On Windows, you need a terminal emulation program such as PuTTY. On Chromebooks, you need a terminal program such as Beagle Term.

If you're using a Chromebook, plug in your Circuit Playground Express, and then run Beagle Term. You should get a screen similar to the one in Figure 6-12 with a `/dev/ttyACM0` or `ACM1` port already filled in. If you need to switch ports to a different USB device, do it here. Then click the Connect button (the other settings are fine).

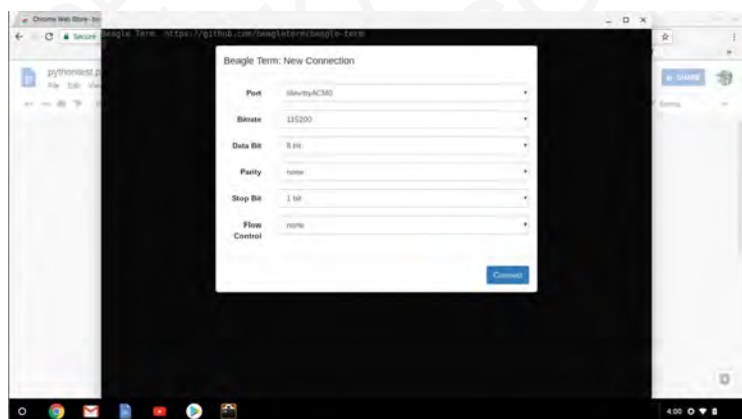


FIGURE 6-12. Beagle Term on Chromebook, settings screen

Figure 6-13 shows the corresponding Beagle Term output from the CircuitPython `print` statement.

Once you have a method for getting data from Circuit Playground Express to your computer, what data might be output?

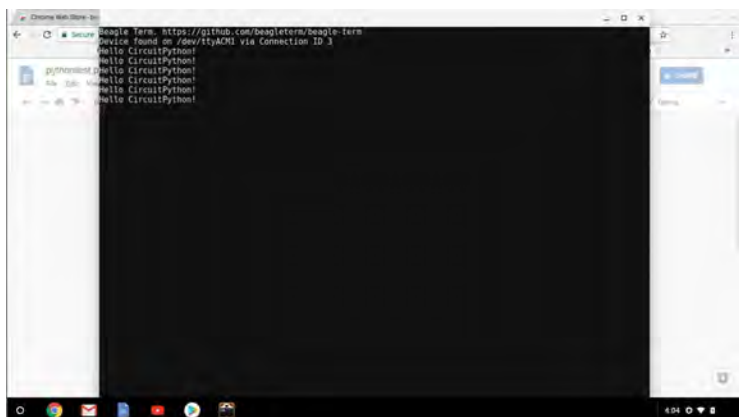


FIGURE 6-13. Chromebook serial output to Beagle Term for the Hello CircuitPython demo

With all the sensors on the board, it would be great to use Circuit Playground Express to measure things like temperature and light intensity and send them back to the computer to record for later use. The larger NeoPixel LED lights can indicate certain values measured by the board, but actual numbers for values makes the output easier to use for measurements. You can do so easily (and you will in the next chapter) when you learn more about functions available for program use.

The Circuit Playground Express code library has the functions that let you use all the capabilities of the board. The next section will list them all for reference.

THE ADAFRUIT CIRCUIT PLAYGROUND EXPRESS LIBRARY

To effectively use the capabilities of Circuit Playground Express in CircuitPython, we need a reference for all the library functions available. This section lists the functions available as of this writing. An up-to-date list (for example, if Adafruit changes or adds to the library) is located at <http://circuitpython.readthedocs.io/projects/circuitplayground/en/latest/api.html>.

All the code fragments in Table 6-1 assume you have the Python import statement at the top of your code like this:

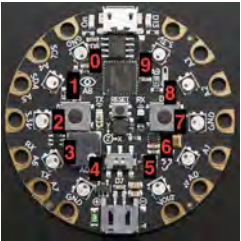

```
from adafruit_circuitplayground.express import cpx
```

NOTE A *parameter* is a value that you provide to a function to set the behavior of that function. For example, in the two statements `print(3.0)` and `print(x)` the value `3.0.0` and the variable `x` are considered the parameters used for the `print` function.

TABLE 6-1. Adafruit Circuit Playground Express library functions and usage

ACTION	CPX LIBRARY USAGE
Accelerometer	<pre>x, y, z = cpx.acceleration print((x, y, z))</pre>
Button A	<pre>if cpx.button_a: print("Button A pressed!")</pre>
Button B	<pre>if cpx.button_b: print("Button B pressed!")</pre>
Slide Switch	<pre>print("Slide switch:", cpx.switch) (True to the left, False to the right)</pre>
Red (D13) LED	<pre>cpx.red_led = True time.sleep(1) cpx.red_led = False time.sleep(1) (True turns LED on, False turns LED off)</pre>
Temperature Sensor Value	<pre>temperature_c = cpx.temperature temperature_f = temperature_c * 1.8 + 32 print("Temperature celsius:", temperature_c) print("Temperature fahrenheit:", temperature_f)</pre>

ACTION	CPX LIBRARY USAGE
Light Sensor Value	<code>print("Light Value:", cpx.light)</code>
Play WAV file	<code>cpx.play_file("laugh.wav")</code> The WAV file must be on the board flash drive.
Play a tone on the speaker (fixed duration)	Parameters: <i>frequency (integer)</i> —The frequency of the tone in Hz <i>duration (decimal)</i> —The duration of the tone in seconds <code>cpx.play_tone(440, 1.0) # 440 hz for 1 second</code>
Play a tone on the speaker (until told to stop)	Parameter: <i>frequency (integer)</i> —The frequency of the tone in Hz <code>cpx.start_tone(262)</code>
Stop playing a tone previously started with <code>start_tone</code>	<code>cpx.stop_tone()</code>
Detect the board being tapped	<code>cpx.detect_taps = 1</code> <code>if cpx.tapped:</code> <code>print("Single tap detected!")</code>
Detect the board being double-tapped	<code>cpx.detect_taps = 2</code> <code>if cpx.tapped:</code> <code>print("A double-tap detected!")</code>
Detect when board is shaken	Parameter: <i>shake_threshold (integer)</i> —The threshold shake must exceed to return True (default: 30). Lower = more sensitive; keep above 10. <code>if cpx.shake():</code> <code>print("Shake detected!")</code> <code>if cpx.shake(100):</code> <code>print("Hard shake detected!")</code>
Set NeoPixel LEDs	Set the color of NeoPixels; values are Red, Green, Blue, and can each range from 0 (off) to 255 (full on): <code>cpx.pixels[9] = (30, 0, 0)</code>

ACTION	CPX LIBRARY USAGE
Set NeoPixel LEDs (continued)	<p>Pixels are numbered counterclockwise from <code>cpx.pixels[0]</code> through <code>cpx.pixels[9]</code>.</p>  <p>You can also use a hexadecimal value in the format <code>0xRRGGBB</code> (decimal 30 = hex <code>0x1e</code>):</p> <pre>cpx.pixels[9] = 0x1e0000</pre> <p>All the pixels can be lit to the same color value specified using <code>cpx.pixels.fill</code>:</p> <pre>cpx.pixels.fill((30, 0, 0))</pre> <p>To turn off all pixels, set them to <code>(0, 0, 0)</code>:</p> <pre>cpx.pixels.fill((0, 0, 0))</pre>
Set NeoPixel brightness	<p>Set the brightness of all pixels (from <code>0.0</code> to <code>1.0</code>):</p> <pre>cpx.pixels.brightness = 0.3</pre>
Pads A1 through A7 being touched	<p>The touch pads are around the edge of the board.</p>  <pre>if cpx.touch_A1: print('Touched pad A1')</pre> <p>Change A1 to A2, A3, A4, A5, A6, A7 for other pads.</p>

ACTION	CPX LIBRARY USAGE
Set touch pad sensitivity	Parameter: <i>adjustment (integer)</i> —The desired threshold increase; higher numbers make the touch pads less sensitive. <pre> cpx.adjust_touch_threshold(200) while True: if cpx.touch_A1: print('Touched pad A1 hard') </pre>

CircuitPython API Documentation

You can get the information for using the libraries built to implement Circuit Playground Express–specific hardware control functions at <http://circuitpython.readthedocs.io/>.

The code that makes up how our program communicates with a predefined set of code written by another group is often called an applications programming interface (API). If you like, you can look at the source code for Circuit Playground Express library functions in the Adafruit GitHub repository at https://github.com/adafruit/Adafruit_CircuitPython_CircuitPlayground/blob/master/adafruit_circuitplayground/express.py.

Go ahead and try some of these functions in your own code. To get ideas on how others are using CircuitPython on Circuit Playground Express, visit <https://learn.adafruit.com/category/express> and look for projects that are using CircuitPython.

RUNNING CODE ON EXPRESS VIA THE REPL

Earlier in the chapter, we used the Serial window in Mu (or Beagle Term for Chromebook) to view the output from a CircuitPython file. Within the Serial window, you can press Ctrl-C to get to the REPL. The REPL is an interactive method for entering commands into CircuitPython and getting feedback.

First, connect your Circuit Playground Express board to your computer with a USB cable. Run the Mu editor (Windows/Mac) or Beagle Term (Chromebook).

If all is good, you will see the editor window shown in Figure 6-14. Click the REPL button, which has a keyboard icon.

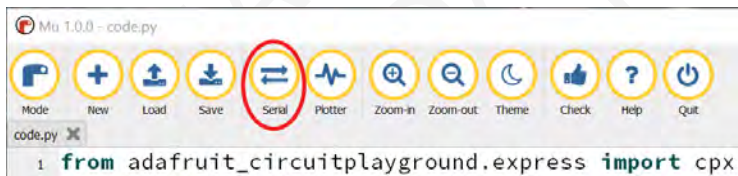


FIGURE 6-14. Start the Serial by clicking the Serial icon at the top of the Mu screen and press Ctrl-C to get the >>> prompt.

The editor window will split in half (Figure 6-15). The REPL is in the bottom portion.

The Serial window will show your serial output/input. But it will also communicate with the board. If you press Ctrl-D, the code will start again without doing a full board reset (which pressing the button onboard does).

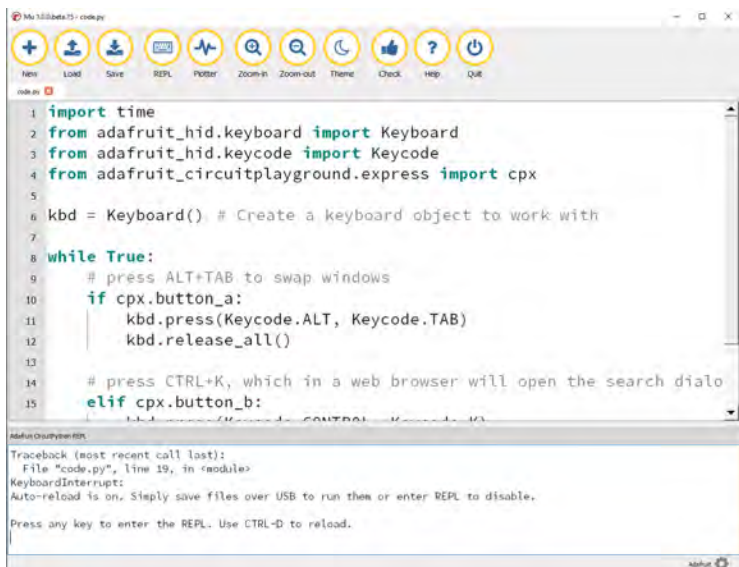


FIGURE 6-15. The Serial window at the bottom of Mu

If you press any other keyboard key, you enter the REPL itself (Figure 6-16). Now any commands you type into the window will be interpreted as CircuitPython commands.

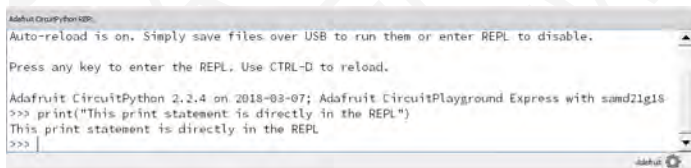


FIGURE 6-16. Typing CircuitPython commands into the REPL

If you would like to import a library (Figure 6-17), you can do so first and then you can use library functions after that.

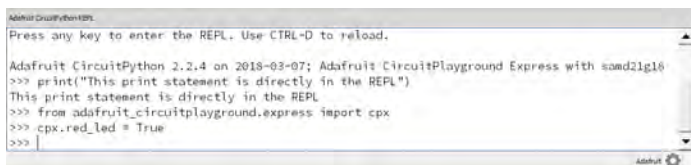


FIGURE 6-17. Importing a library in the REPL

The code you type is interactive, but there is no mechanism to save your code. This is why writing code in a text editor is best for code you will be changing.

WRAP-UP

CircuitPython provides a great way to program Circuit Playground Express using a programming language gaining in popularity. CircuitPython also exposes additional functionality on Circuit Playground Express, including the ability to read and write files placed on the onboard flash memory.

You can also type CircuitPython commands into the REPL if you need to perform a short list of actions.

In the next chapter, some of the more advanced uses of CircuitPython for Circuit Playground Express will be covered.

CHAPTER QUESTIONS

1. What is an interpreted computer language?
2. In MakeCode, the web interface places a binary UF2 file onto Circuit Playground Express when the download button is used. How is code placed onto Circuit Playground Express in CircuitPython, and where is the mechanism to turn code into binary machine commands?
3. What is the CircuitPython equivalent command to the MakeCode `forever` loop?
4. Which CircuitPython function outputs information from Circuit Playground Express to the connected computer?

A



Troubleshooting

Running into problems and solving them is a defining part of the Maker experience. This appendix will help you resolve many common issues you may face when working with Circuit Playground Express.

Most issues fall into the following categories:

- * Cable issues
- * Connectivity issues
- * Software issues
- * Common library problems
- * Error messages
- * Usage issues
- * Manufacturer support

USB CABLE AND POWER ISSUES

NOTE Many Circuit Playground Express issues are ultimately traced to a bad USB cable or a power issue.

To get an older USB Micro-B cable, you may scrounge in a box of old cables to find something that works. This approach does not always get you the reliable cable you need. Problems you may encounter include the followings:

- ✱ Some USB Micro-B cables have only power wires and no signal wires (they were designed for charging devices only).
- ✱ The cable connections are broken or intermittent, due to flexing (often at one end or the other). This can happen if the cable was heavily used.
- ✱ The wire gauge of the cable is insufficient (an uncommon issue, but it might happen with smaller or more inexpensive cables, often sourced from discount suppliers).
- ✱ A connector is cracked, dirty, or broken inside.

With power, be sure the green “On” LED is steadily green at all times. With data, you can go into Microsoft MakeCode and create a small test program and see whether Circuit Playground Express will load the code and execute it. Using MakeCode is simplest at this point since it requires no external software other than a web browser and the website <https://makecode.adafruit.com/>. If you do not have Internet connectivity, you can code simple programs in CircuitPython or Arduino to accomplish a similar result.

You may think, “This cable works for my phone—it should be good.” However, the phone may not use the data wires per standard USB specifications, or it may have only power wires. That the

cable works for a phone is not a sufficient indication that the cable will work 100 percent in Circuit Playground projects.

Here are some power troubleshooting steps:

1. Check your connections and USB port to make sure that everything connects well.
2. If there is a problem, try swapping the cable for a thicker, more substantial one, or consider purchasing a new one.
3. As a final check, disconnect the USB cable and connect your Circuit Playground Express to external power. You have three choices:
 - * Connect a charged LiPo battery to the JST battery connector opposite the USB connector.
 - * Use a “Phone Rescue” battery, the type that uses a rechargeable battery and a USB-to-micro-B cable, often to provide extra power to a mobile phone or tablet.
 - * Adafruit sells a battery pack taking three AAA cells and provides a battery connector suitable for Circuit Playground Express (Adafruit product # 727). Be sure the on-off switch is in the on position.

Buying a good, substantial cable (Adafruit #2008 or similar) from a local shop or reputable online supplier will remedy many issues.

If at this point you have tried to power the board using multiple methods and the power On LED will not glow green and the board appears dead, see the section “Manufacturer Support,” later in this appendix.

If you plugged the board in and there was a flash and now it appears dead, wait about 10 minutes and try again with all external connections removed. There could have been a short circuit if metal touched bridged pads on the bottom of the board.

At this point, if you apply proper power and you do not get a green On LED next to the USB port, Circuit Playground Express may be “dead.” It can happen to electronics, especially if they are treated poorly. If your Circuit Playground Express is new, contact the manufacturer. If the board has been working for a while and you know you did something to make it no longer work, you may need to get another Circuit Playground Express. It happens at times—it is part of the experimentation process.

CONNECTIVITY ISSUES

Circuit Playground Express may have problems talking to a larger computer used to program the device. First review the “USB Cable and Power Issues” section prior to diagnosing connectivity issues to ensure the problem is not power or cable related.

Problems with connectivity include the following:

- * Intermittent communications on USB 3 ports on computers (USB 3 connectors often have blue plastic inside them)
- * Compatibility issues on USB ports on some versions of the Linux operating system
- * USB ports not recognizing Circuit Playground Express

Connectivity problems generally do not result in error messages. Look at these possible situations:

General communications: Is your USB cable connected to a USB 3 port?

Reconnect your Circuit Playground Express to a USB 2 port if you have one available. If you are using a USB 2 hub, try to plug into the main port and not the hub. If you use a hub, a powered hub would be better to ensure the current available is enough for your project.

I get the green power LED, but my Circuit Playground Express appears to not communicate in any way; my program is not loaded.

Check the “USB Cable and Power Issues” section.

I cannot find Circuit Playground Express in the list of devices in Windows.

In Windows, Circuit Playground Express shows up under the “Unspecified” category of devices (Figure A-1). In Device Manager, it is under Ports (COM & LPT) as a USB Serial Device, with the Windows communications port listed in parentheses (Figure A-2).

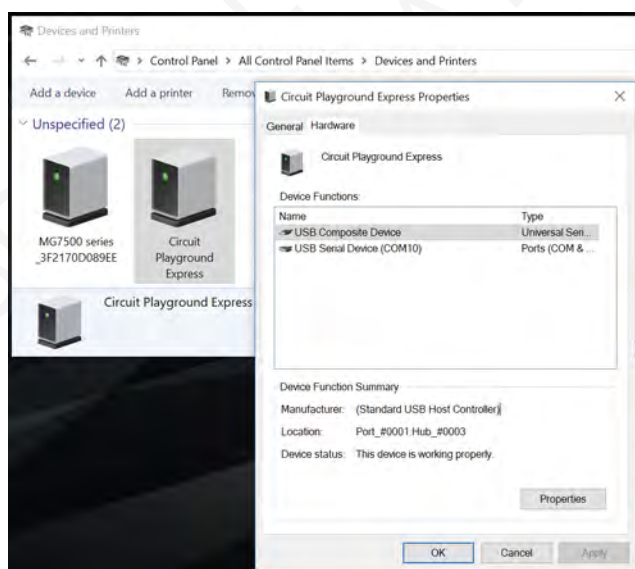


FIGURE A-1. Circuit Playground Express listed in Windows 10 Devices and Printers section of Control Panel

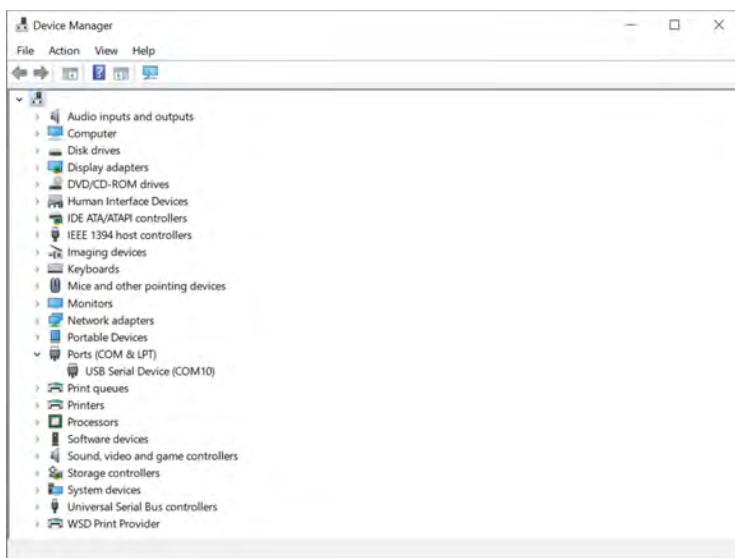


FIGURE A-2. Windows Control Panel Device Manager listing Circuit Playground Express as a USB Serial Device (COM10 here)

The COM port may be numbered differently depending on the port your Circuit Playground Express is plugged into.

I'm using VMware [or another virtual machine program] and I'm having issues.

Some VM programs have problems connecting to “real” computer USB ports. This is not a problem limited to Circuit Playground Express but applies to any USB device. See the VMware or other software forums for USB-specific advice.

Both Microsoft MakeCode and CircuitPython will not work with my Circuit Playground.

Check that you have Circuit Playground *Express*. The difference between the Express and the Circuit Playground Classic

is discussed in Chapter 1. Circuit Playground Classic does not run MakeCode or CircuitPython.

My Circuit Playground Express worked when I first got it, but it is acting up now. What could be the problem?

First, check your power connections; if they are not good, correct them. Next, if you have connected external components, your circuit could be electrically problematic or miswired. Remove any connections and try to load a basic Blink sketch in one of the programming languages to test it out. If it works outside your project, check your project connections. If the power On light does not come on, check the “USB Cable and Power Issues” section.

Can I charge a LiPo rechargeable battery to power my Circuit Playground Express?

You can use a LiPo battery for Circuit Playground Express, but note the board cannot recharge the LiPo if it’s plugged into a USB port. Unplug the battery from your Circuit Playground Express and charge the battery with a circuit board specifically designed to recharge the battery safely. Adafruit sells several types of LiPo recharging boards; product #s 1304 and 1904 work well at a low cost. The size of the LiPo battery will determine how long the battery will last—the larger the battery, the longer it will power the board. You can save power in a project by using NeoPixels sparingly. Consider using the slide switch to programmatically “turn off” the NeoPixels.

CIRCUITPYTHON ISSUES

I edit my program and copy it to the CIRCUITPY drive but Circuit Playground Express doesn't behave as if it recognizes the code.

The CircuitPython file you copy over to run on Circuit Playground Express should always be called `code.py`. This is so that the Python interpreter knows the name of the code you want to run.

If you use a name like `mycoolprog.py`, the board will not know that file is the code you wish to run. Feel free to use a more descriptive name on your backup storage device copies—for example, `music-on-tilt.py`.

There are four options for filenames for the code the board will run: `code.txt`, `code.py`, `main.txt`, and `main.py`. CircuitPython looks for those files, in that order, and then runs the first one it finds.

Adafruit highly suggests that you use the filename `code.py`.

If your program doesn't seem to be updating as you work, make sure you haven't created another code file that's being read instead of the one you're working on.

My CircuitPython program cannot find the libraries/modules it needs to work.

CircuitPython looks for library code in the subdirectory `/lib` on the onboard flash drive. Go to Chapter 6 to review the process to copy all of the Adafruit CircuitPython libraries into `/lib`. The libraries use approximately 400KB of space. Even with all Adafruit libraries loaded into `/lib`, there should be plenty of room for other code on the CIRCUITPY drive. See Chapter 6 for more details on installing the Adafruit CircuitPython libraries.

I plug in the Circuit Playground Express and I get a CIRCUITPY drive. But I cannot get the CircuitPython code.py to copy over or run when it should.

Double-check if you are using the modified `boot.py` for writing files listed in Chapter 7. If that code is running, first move the slide switch back to file mode. You can then rename `boot.py` to **`boot-py.old`** on the CIRCUITPY drive and press Reset to get the flash drive back to standard operation.

If you have a file on your Circuit Playground Express named `code.txt`, run it instead of `code.py`. Rename or delete `code.txt` and ensure your code is named `code.py`.

Finally, in a rare event, the flash chip may have an issue. Follow these steps to erase the flash chip and enable normal operation:

1. Type this short program and save it to the CIRCUITPY drive as `code.py`:

```
import storage
storage.erase_filesystem()
```
2. Click the Reset button on the board to ensure the code is run.
3. Now see if you can copy a CircuitPython program of your choice to the board and have it run.

If you are still having issues, follow the instructions in Chapter 6 on reinstalling the latest version of CircuitPython.

ARDUINO IDE ISSUES

At this point you have gone through the connectivity issues, and everything seems to be working. But you appear to be having errors in the Arduino IDE, either during the compile/verify stage or during upload.

I get several errors when I try to upload a program from the Arduino IDE.

Make sure you've selected Circuit Playground Express in the Tools → Board menu and have selected the proper serial port in Tools → Port. If you switch back to another Arduino-compatible board, change the settings appropriately.

COMMON ARDUINO LIBRARY PROBLEMS

There are many problems you can have with using libraries. The most common library-related error messages take the form “XXXX does not name a type” or “YYYY not declared in this scope.” They mean that the compiler could not find the library. This can be due to any of the following causes:

The library is not installed.

See the steps in Chapter 6 on how to install a library correctly.

Arduino cannot find the library folder.

The IDE will find standard libraries and libraries installed only in the sketch `libraries` folder.

The specific library folder must be at the top level of the `libraries` folder. If you put it in a subfolder, the IDE will not find it.

You do not have a “Sketchbook” folder.

It is there, but on a Windows or macOS machine it is named `Arduino` (on Linux it is named `Sketchbook`).

A library is incomplete.

You must download and install the entire library. Do not omit or alter the names of any files inside the `libraries` folder.

A folder name is wrong.

The IDE will not load files with certain characters in the name. Unfortunately, it does not like the dashes in the ZIP filenames generated by GitHub. When you unzip the file, rename the folder so that it does not contain any illegal characters. Simply replacing each dash (-) with an underscore (_) usually works. If the folder has the word `master` on the end (usually preceded by a dash), remove that also. The best method to see what the library name should be is to look at the sample code to see what the sample expects the library name to look like.

The library name is spelled incorrectly.

The name specified in the `#include` line of your sketch must match exactly (including capitalization!) the name in the library. If it does not match exactly, the IDE will not be able to find it. The example sketches included with the library will have the correct spelling. Just cut and paste from there to avoid typographical errors.

You have a wrong version of a library or multiple copies of the same library in accessible folders.

If you have multiple versions of a library, the IDE will try to load all of them. This will result in compiler errors. It is not enough to simply rename the library folder; it must be moved outside of the sketchbook `libraries` folder so the IDE will not try to load it.

One library depends on another library.

Some libraries are dependent on other libraries. For example, most of the Adafruit graphic display libraries are dependent on the Adafruit-GFX library. You must have the GFX library installed to use the dependent libraries. This is true

as well for libraries that use I2C that also expect the Wire library.

The IDE needs to be restarted.

The IDE searches for libraries at startup. You must shut down *all* running copies of the IDE and restart before it will recognize a newly installed library.

I found a wonderful Arduino library that does what I need, but when I try to use it on my Circuit Playground Express, I get errors. What can I do?

The library you found was probably coded for other microcontrollers. Those libraries might use large memory spaces or hardware in other microcontrollers, which may not work on Circuit Playground Express. If you understand how the library code works, you may be able to fix some errors yourself. Performing a Google search for the library name may produce pages where others encountered the same circumstance and recoded the library.

Does a library I found on the Internet work with Circuit Playground Express?

Because there are hundreds of libraries out there written by all sorts of people, it may or may not work. Many libraries expect an Arduino Uno and not the larger processor on Circuit Playground Express. But it doesn't hurt to try—see the previous question to proceed.

ERROR MESSAGES

Error messages may fall into the general categories listed here.

Arduino Compilation Issues

Advanced: Can I write code that will compile one way for Circuit Playground and another for Circuit Playground Express in Arduino?

Yes, the Arduino IDE internal preprocessor provides separate definitions for the boards that can be tested. Circuit Playground Classic can be tested using AVR; see the following sample code:

```
#ifndef __AVR__      // Circuit Playground 'classic'
#include "utility/CPlay_CapacitiveSensor.h"
#else
#include "utility/Adafruit_CPlay_FreeTouch.h"
#include "utility/IRLibCPE.h"
#endif
```

Arduino Upload Errors

Be sure you have done the following:

1. Ensure you have a known good USB cable with both power and data lines.
2. Set the Tools → Board menu to Circuit Playground Express.
3. Ensure Tools → Port is set to the communications port that your operating system assigns when you plug in the board. Often the port will say “Circuit Playground Express” next to it, but it might not if things are being balky.
4. If there are still issues, press the Reset button to see if the Arduino IDE will recognize the board.

The Arduino Serial Monitor

While using Arduino, you should use the `Serial.print` and `Serial.println` functions to provide feedback in the Arduino serial monitor as to what the board is doing while coding and debugging. Select Tools → Serial Monitor to see the output (unlike Mu,

the output goes to a separate window that you must specifically open after the program is running). The `Serial.print` statements can later be commented out for a final “ready to use” program.

USAGE ISSUES

You may encounter the following issues while using Circuit Playground Express.

Can other pads besides A1, A2, A3, A4, A5, A6, and A7 do capacitive touch? I would like to have more capacitive touch inputs.

Unfortunately, those are the only pads that work with capacitive touch. A0 does not “do” touch, and neither do the other pads. Consider using two Circuit Playground Express boards to add to the number of touchpads. Adafruit sells capacitive touch expansion boards, but the coding would be complex. It’s best to use multiple Circuit Playground Express boards.

Windows 7 (or Windows 8) is not recognizing the board.

See Chapter 2 to learn how to install drivers for Windows 7 and 8. Windows 10, macOS, and Linux do not need drivers.

I would like to try using the Arduino IDE in Linux. What are the pitfalls I need to look out for?

The software may need access to the USB port, but this is controlled by root. You may need to set the USB port for dial-out. Check the Adafruit support forums for Linux issues specific to Circuit Playground Express at <https://forums.adafruit.com/viewforum.php?f=58>.

Will a Circuit Playground Express interface to the hardware I have?

The answer is possibly. Two factors are involved: voltage compatibility and software support.

The input and output pads for Circuit Playground Express are 3.3 volts. The external circuitry should work with a digital output of 3.3 volts. External circuitry should *never* put more than 3.3 volts on an input/output pin because this might damage the board (5 volts on the USB connector is fine, though).

Depending on the function of an external circuit, code will be required to make the circuit function. Sometimes code is easy, or it could be quite complex. It is beyond the scope of this Getting Started book to discuss all the external circuits that can be connected and programmed with the board. You may have to experiment and read up on the subject in other resources.

Are the Circuit Playground Express EAGLE CAD circuit board (PCB) layout files available?

Yes; see <https://github.com/adafruit/Adafruit-Circuit-Playground-Express-PCB>.

MANUFACTURER SUPPORT

Adafruit Industries makes customer service and satisfaction a cornerstone of its business. If you still have problems after troubleshooting, you can visit the Adafruit forums (<https://forums.adafruit.com/>) to describe your situation. The helpful forum moderators will be able to assist with additional troubleshooting.

You'll also find many tutorials on using Circuit Playground Express and other Adafruit products at <https://learn.adafruit.com/>.

After posting to the Adafruit forum, if it is evident your board is defective, Adafruit may replace it (at their discretion). Treat your electronics with care and they should last nearly forever. Just don't spill your drink on it or take it to Burning Man, and then suspect it was a factory fault.

SPECIAL
HACKADAY
SUPERCON
PREVIEW

B

.....

Reference Materials

The main subjects of this book—how to write code in Microsoft MakeCode, how to use CircuitPython, and how to use the Arduino IDE—could each easily be the basis for its own full-length book. In this Getting Started series book, we explored each subject in the space available.

In the following sections, other resources for information are listed for further study. Also consider that new information on the subjects in this book will be published after this book goes to print. Using a search engine of your choice can help if you have exhausted the information in this book and the references that follow.

ON THE INTERNET

The Internet provides a wealth of information. All of the references noted are free to view. Adafruit Industries materials are generally licensed so that you can use the materials any way you want (with attribution).

Circuit Playground Express

- * Adafruit Circuit Playground Express Guide: <https://learn.adafruit.com/adafruit-circuit-playground-express>
- * Adafruit Customer Support forums: <https://forums.adafruit.com/>

Microsoft MakeCode

- * Microsoft MakeCode for Circuit Playground Express: <https://makecode.adafruit.com>
- * The main Microsoft MakeCode site: <https://makecode.com>
- * Adafruit Learn Microsoft MakeCode: <https://learn.adafruit.com/makecode>
- * Information on the UF2 file format: <https://github.com/microsoft/uf2>

Python and CircuitPython

- * Adafruit Welcome to CircuitPython! <https://learn.adafruit.com/welcome-to-circuitpython>
- * Adafruit CircuitPython Essentials: <https://learn.adafruit.com/circuitpython-essentials>
- * The Python Software Foundation, Python for Beginners: www.python.org/about/gettingstarted/
- * The Beginners Guide for Programmers: <https://wiki.python.org/moin/BeginnersGuide/Programmers>
- * CircuitPython API Reference: <http://circuitpython.readthedocs.io/en/latest/>
- * A list of CircuitPython resources maintained by Adafruit: <https://github.com/adafruit/awesome-circuitpython>

Arduino

- * Primary Arduino website: www.arduino.cc
- * Arduino Language Reference: www.arduino.cc/reference/en/
- * Arduino Tutorials: www.arduino.cc/en/Tutorial/
- * C Tutorial: www.tutorialspoint.com/cprogramming/index.htm
- * Adafruit Circuit Playground Express Guide: <https://learn.adafruit.com/adafruit-circuit-playground-express>
- * Adafruit Ladyada's Learn Arduino: <https://learn.adafruit.com/ladyadas-learn-arduino-lesson-number-0>

Chrome OS

- * Caret Text Editor: <https://chrome.google.com/webstore/detail/caret/fljalecfjciodhpcledpamjachpml?hl=en>
- * Beagle Term Terminal Emulator: <https://chrome.google.com/webstore/detail/beagle-term/gkdofhllgfohlldimiildbgogdpoea?hl=en>
- * YouTube video on using a Chromebook with Circuit Playground Express: www.youtube.com/watch?v=B-PfKv7DCbc

PUBLICATIONS

The following resources may help you learn some of the concepts in this book:

- * *Getting Started with Arduino, Second Edition*, by Massimo Banzi (co-creator of Arduino)
- * *Programming Arduino: Getting Started with Sketches*, by Simon Monk
- * Once you've grasped the basics of setting up the Arduino IDE, check out books and other resources on creating

projects based on Arduino. Browsing your favorite technical bookstore will provide a wide range of books and magazines.

- * There are many books on learning Python, but it is very difficult to recommend any one title as a companion for a Getting Started book. Most Python books are written for more experienced programmers or extensively use concepts such as object-oriented language constructs unnecessary for beginners.

SPECIAL
HACKADAY
SUPERCAN
PREVIEW

About the Author

Engineer and Maker Mike Barela is currently a consultant for Adafruit Industries, LLC. He recently retired as a senior Foreign Service officer and security engineer for the U.S. Department of State. Mike is a graduate of Whitman College (mathematics/physics) and the California Institute of Technology (electrical engineering). He has also worked at Hewlett-Packard, the Caltech/NASA Jet Propulsion Laboratory, and Boeing. An avid electronics enthusiast, he started with a workbench and Radio Shack parts in high school. Mike is the author of the book *Make: Getting Started with Adafruit Trinket* as well as tutorials on the Adafruit Learning System at <https://learn.adafruit.com/>.

To purchase this book in its entirety, please visit

<https://amzn.to/2CMD3vZ>