adafruit learning system

Digital Fidget Spinner

Created by Tony DiCola

Last updated on 2018-08-22 04:00:56 PM UTC

Guide Contents

Overview

Fidgets spinners are trendy gadgets that smoothly spin around at the lightest touch. You can 3D print a fidget spinner (https://adafru.it/wVF), mill one from acrylic (https://adafru.it/wWa), or even buy one to dye with hydro-dipping (https://adafru.it/wWb) to have a custom and unique fidget spinner. However what if you could have an even more unique fidget spinner, one with *no moving parts*? The digital fidget spinner with Circuit Playground is just that--a fun fidget spinner with no moving parts that spins NeoPixel lights when flicked or tapped!

This project is built with Circuit Playground (either the classic (https://adafru.it/ncE) or new express (https://adafru.it/wpF) version) which is an all-in-one electronics learning board. Everything needed to build this project is built-in to the Circuit Playground board so you can get started very easily without any soldering or assembly required. An accelerometer at the center of Circuit Playground serves as the sensor to detect when the board is tapped or flicked. Once the board is flicked it moves the ring of 10 NeoPixels around the outside in a fun, colorful fidget spinner animation. The best part is since it's only software controlling this spinner you can customize the colors, animation, and more by just changing the code!

There are three ways to build this project too, either as an Arduino-based sketch that works with Circuit Playground classic and express, as a CircuitPython-based project that works just with Circuit Playground express, or as a Microsoft MakeCode project that uses Circuit Playground express. All the versions have similar behavior and are a good demonstration of what's possible with Arduino, CircuitPython/MicroPython, and Circuit Playground boards.

Before you get started you'll want to familiarize yourself with Circuit Playground boards by reading the Circuit Playground classic guide (https://adafru.it/ncG). In addition if you're building the CircuitPython version check out the MicroPython (what CircuitPython is derived from) guides on the learning system (https://adafru.it/wWc). And if you're building the MakeCode version be sure to read the MakeCode guide (https://adafru.it/xCZ).

Hardware

The hardware for this project is very simple, you only need a **Circuit Playground Classic (https://adafru.it/ncE)** or **Circuit Playground Express (https://adafru.it/wpF)** board! Everything necessary to make the fidget spinner is built in to the Circuit Playground board.

If you're planning to build the Arduino version of this project you can use either the **Circuit Playground Classic (https://adafru.it/ncE)** or **Circuit Playground Express (https://adafru.it/wpF)** board (the express board is newer and has a fancier microcontroller with more memory and features compared to the classic board).

If you're planning to build the CircuitPython version of this project you can **only** use the **Circuit Playground Express (https://adafru.it/wpF)** board. The older classic board won't work because it isn't powerful enough to run Circuit Python code.

Before you get started make sure to follow the guide for your board to check you can program and load code onto it (in particular you might need to install drivers for some platforms like Windows):

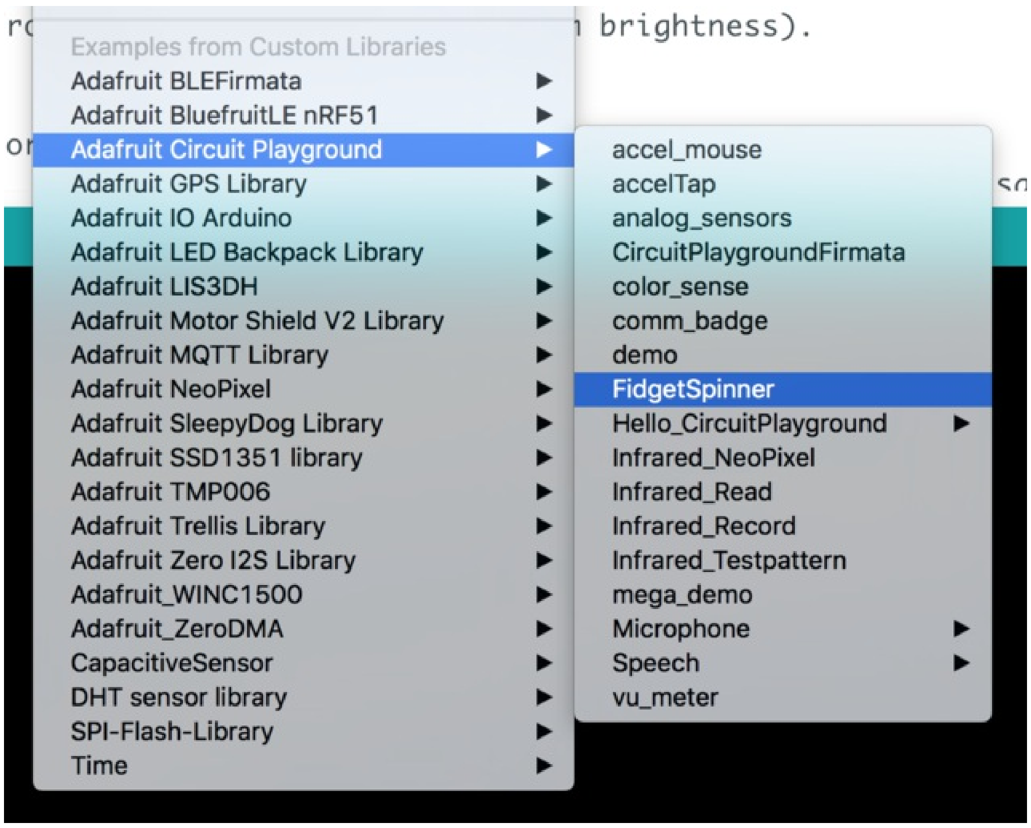
- Circuit Playground Classic guide (https://adafru.it/ncG)
- Circuit Playground Express guide (https://adafru.it/wWd)

Arduino

The Arduino version of the digital fidget spinner is a good example of performing peak detection with a signal. In this case the signal is one of the axes of the accelerometer and a sudden peak, or tap/flick of the board, causes the LEDs around the board to animate with movement. By applying a decay to the velocity of the LED animation the spinner will slow down as if it were affected by friction and air resistance.

To use this example you'll need to install the Adafruit Circuit Playground library in the Arduino IDE. Follow the Circuit Playground guide page on installing the library (https://adafru.it/pAQ). If you've already installed the Circuit Playground library make sure to use the Arduino board manager to check for updates and install the latest version of the library.

Look for a **FidgetSpinner** example in the **Adafruit Circuit Playground library** (you can also get the sketch directly from GitHub here (https://adafru.it/wWe)):



The sketch is ready to run however you might want to change one of the configuration values at the top. Look for the **AXIS** define:

```
// Configure which accelerometer axis to check:
// This should be one of the following values (uncomment only ONE):
//#define AXIS            CircuitPlayground.motionX()    // X axis, good for Circuit Playground express
#define AXIS              CircuitPlayground.motionY()    // Y axis, good for Circuit Playground classic
//#define AXIS            CircuitPlayground.motionZ()    // Z axis, wacky--try it!
```

Notice in the comments it mentions that you can change which accelerometer axis is used to detect a tap of the spinner. For **Circuit Playground Classic** the sketch is setup to use the **Y axis** of the accelerometer which means you should tap the board from the left or right side of the USB & charging ports. However if you're using **Circuit**

Playground Express you'll want to change the AXIS define to use the **X axis** as the accelerometer is in a different position on the board. Comment and uncomment the appropriate line for the board you're using (or experiment by trying the different axes!).

Now upload the sketch to your Circuit Playground board. When the sketch finishes uploading it should start with a red dot on the board LEDs. Try flicking or tapping a side of the board to see the spinner start moving the LED around the board. You can press either of the push buttons to change the display of the spinner:

- One of the buttons (left on Circuit Playground Classic, A on Circuit Playground Express) cycles through four different animations:
    - **Single dot**
    - **Double dots**
    - **Single smooth pulse**
    - **Double smooth pulse**
- The other button (right on Circuit Playground Classic, B on Circuit Playground Express) cycles through six different color combinations:
    - **Red & black**
    - **Green & black**
    - **Blue & black**
    - **Red & green**
    - **Red & blue**
    - **Green & blue**

If you're curious there are a few other configuration values you can experiment with changing at the top of the sketch:

```
// Configure how the axis direction compares to pixel movement direction:
#define INVERT_AXIS         true  // Set to true or 1 to invert the axis direction vs.
                                  // pixel movement direction, or false/0 to disable.
                                  // If the pixels spin in the opposite direction of your
                                  // flick all the time then try changing this value.

// Configure pixel brightness.
// Set this to a value from 0 to 255 (minimum to maximum brightness).
#define BRIGHTNESS          255

// Configure peak detector behavior:
#define LAG                 30     // Lag, how large is the buffer of filtered samples.
                                   // Must be an integer value!
#define THRESHOLD           20.0   // Number of standard deviations above average a
                                   // value needs to be to be considered a peak.
#define INFLUENCE            0.1   // Scale down peak values to this percent influence
                                   // when storing them back in the filtered values.
                                   // Should be a value from 0 to 1.0 where smaller
                                   // values mean peaks have less influence.

// Configure spinner decay, i.e. how much it slows down.  This should
// be a value from 0 to 1 where smaller values cause the spinner to
// slow down faster.
#define DECAY                0.66
```

- The **INVERT_AXIS** option flips which direction the LEDs spin when the board is tapped. Depending on how you hold the spinner and the orientation of the accelerometer you might need to set this to true or false to line up the tap direction and LED movement.

- **BRIGHTNESS** controls how bright the LEDs are on the board. The default of 255 is maximum brightness, but you can bump it down to a smaller positive value like 100 for dimmer LEDs.
- **LAG**, **THRESHOLD**, and **INFLUENCE** control how the flick/peak detection works. Read the comments and watch the video at the top of the page to see more about how these values change the behavior of the peak detection. You can make the detection more or less sensitive by changing these values.
- **DECAY** controls how quickly the spinner slows down after it starts spinning. This is a value from 0 to 1.0 where higher values mean the spinner will spin for longer (i.e. less 'friction').

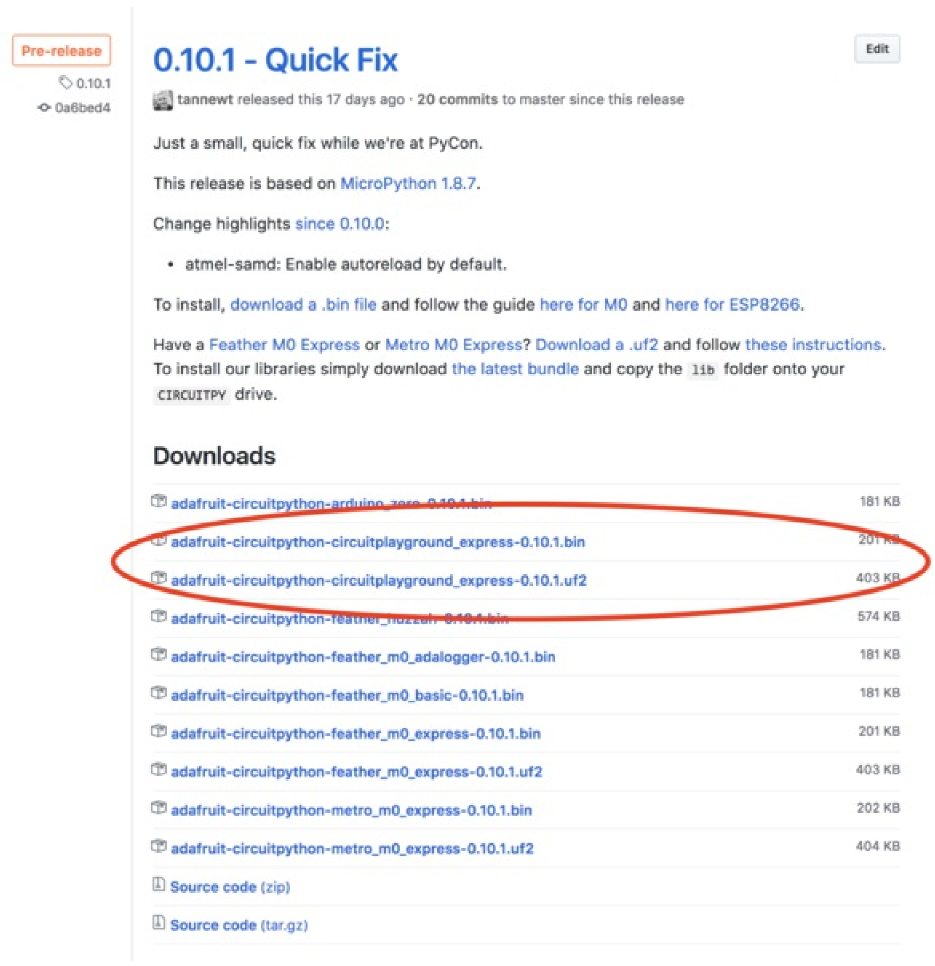
That's all there is to using the Arduino version of the digital fidget spinner!

CircuitPython

With CircuitPython and Circuit Playground Express you can create a similar digital fidget spinner that's powered entirely by Python code! This spinner has the same functionality as the Arduino version but it's written in easier to understand Python code. If you aren't familiar with MicroPython & CircuitPython be sure to read more about it here first (https://adafru.it/pXc).

To create the spinner with Python a slightly different approach is taken compared to the Arduino version. With this Python version the spinner relies on built-in tap detection of the LIS3DH accelerometer on the Circuit Playground Express board. This means the spinner code doesn't need to be as fast as the Arduino code because it isn't looking for a small impulse of acceleration. Instead the spinner code waits for the LIS3DH to detect this small impulse or tap on the X axis (left/right sides of the board) and then starts spinning. In addition the CircuitPython version uses an advanced FIFO buffer built-in to the LIS3DH to go back and detect how strong the flick or tap was against the board. Watch the video at the top of this guide for a deeper exploration of how the Python fidget spinner works and how it compares to the Arduino version.

To use the spinner first make sure your Circuit Playground Express board is loaded with the latest version of CircuitPython. See the similar Feather M0 Express board guide for instructions how to load CircuitPython on your board (https://adafru.it/wbv). The instructions are the same for the Feather M0 Express and Circuit Playground Express boards, but be sure to download the **circuitplayground_express** .uf2 or .bin file from the CircuitPython releases page:



0.10.1 - Quick Fix

tannewt released this 17 days ago · 20 commits to master since this release

Just a small, quick fix while we're at PyCon.

This release is based on MicroPython 1.8.7.

Change highlights since 0.10.0:

- atmel-samd: Enable autoreload by default.

To install, download a .bin file and follow the guide here for M0 and here for ESP8266.

Have a Feather M0 Express or Metro M0 Express? Download a .uf2 and follow these instructions. To install our libraries simply download the latest bundle and copy the `lib` folder onto your `CIRCUITPY` drive.

Downloads

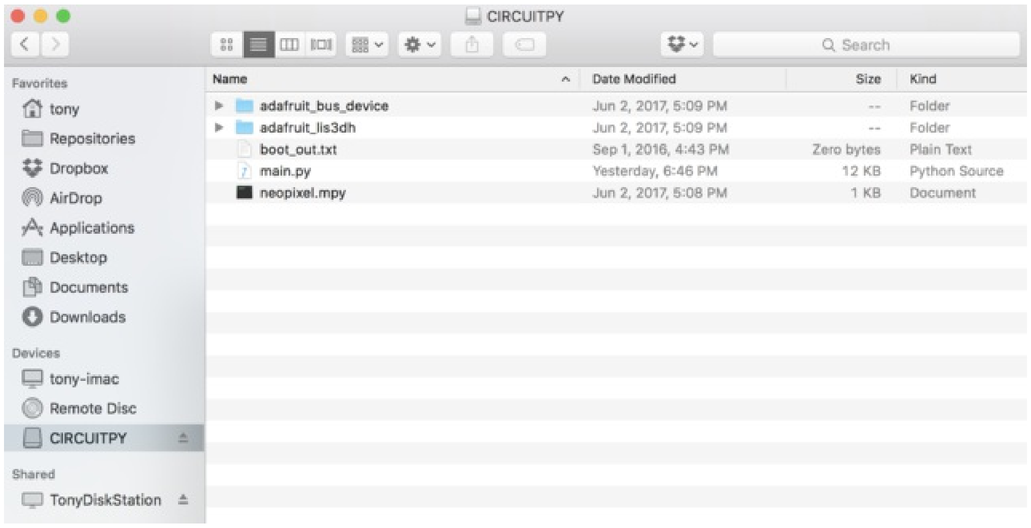
adafruit-circuitpython-arduino_zero-0.10.1.bin	181 KB
adafruit-circuitpython-circuitplayground_express-0.10.1.bin	201 KB
adafruit-circuitpython-circuitplayground_express-0.10.1.uf2	403 KB
adafruit-circuitpython-feather_huzzah-0.10.1.bin	574 KB
adafruit-circuitpython-feather_m0_adalogger-0.10.1.bin	181 KB
adafruit-circuitpython-feather_m0_basic-0.10.1.bin	181 KB
adafruit-circuitpython-feather_m0_express-0.10.1.bin	201 KB
adafruit-circuitpython-feather_m0_express-0.10.1.uf2	403 KB
adafruit-circuitpython-metro_m0_express-0.10.1.bin	202 KB
adafruit-circuitpython-metro_m0_express-0.10.1.uf2	404 KB
Source code (zip)	
Source code (tar.gz)	

Once you download the .uf2 file double tap the Circuit Playground Express board's reset button and drag the .uf2 file to the CPLAYBOOT drive. After a moment the CircuitPython firmware will be flashed to the board.

Next you'll need to copy a few libraries to the board's filesystem. For each of these libraries go to their releases page on GitHub and grab the .zip or .mpy file from the latest release. If the library has a .zip file you'll need to open it and the folder inside is the library, or if it's just a .mpy file that's all you need for the library.

- Adafruit CircuitPython BusDevice library (https://adafru.it/u0d)
- Adafruit CircuitPython LIS3DH library (https://adafru.it/uBt)
- Adafruit CircuitPython NeoPixel library (https://adafru.it/wby)

Remember you can just drag and drop copy these libraries to the **CIRCUITPY** drive your Circuit Playground Express board running CircuitPython will show up as on your computer:



Name	Date Modified	Size	Kind
adafruit_bus_device	Jun 2, 2017, 5:09 PM	--	Folder
adafruit_lis3dh	Jun 2, 2017, 5:09 PM	--	Folder
boot_out.txt	Sep 1, 2016, 4:43 PM	Zero bytes	Plain Text
main.py	Yesterday, 6:46 PM	12 KB	Python Source
neopixel.mpy	Jun 2, 2017, 5:08 PM	1 KB	Document

Finally grab the actual code for the fidget spinner from the examples of the CircuitPython LIS3DH library (https://adafru.it/uBu). There are two files you can use:

- spinner.py (https://adafru.it/wWf) - This is a basic spinner that only spins with one color and animation type. You can see this spinner built in the video at the top of this page.
- spinner_advanced.py (https://adafru.it/wWA) - This is an advanced spinner that changes color and animation type with A and B button presses (just like the Arduino-based spinner). Try this one first if you just want to play with the spinner project.

To run the spinner copy one of the above files to your board and rename it to main.py This will cause CircuitPython to run the code as soon as the board 'boots' up. You should see the board reset itself automatically after copying and renaming the file, but if not press the board's reset button. You should see the spinner start with one red dot. Try flicking or tapping the side of the board (the side with the push buttons) to watch the spinner spin the LED around the board. If you're using the advanced spinner example press either of the A or B buttons to cycle through different colors and animation types (just like the Arduino-based spinner on the previous page).

There are a few configuration values you can change in the spinner code. Open the main.py in a text editor and look for these lines near the top (assuming the spinner_advanced.py above):

```
# Configuration:
ACCEL_RANGE   = adafruit_lis3dh.RANGE_16_G  # Accelerometer range.
TAP_THRESHOLD = 20              # Accelerometer tap threshold.  Higher values
                                # mean you need to tap harder to start a spin.
SPINNER_DECAY = 0.5             # Decay rate for the spinner.  Set to a value
                                # from 0 to 1.0 where lower values mean the
                                # spinner slows down faster.
# Define list of color combinations.  Pressing button A will cycle through
# these combos.  Each tuple entry (line) should be a 2-tuple of 3-tuple RGB
# values (0-255).
COLORS = (
    ((255, 0, 0), (0, 0, 0)),      # Red to black
    ((0, 255, 0), (0, 0, 0)),      # Green to black
    ((0, 0, 255), (0, 0, 0)),      # Blue to black
    ((255, 0, 0), (0, 255, 0)),    # Red to green
    ((255, 0, 0), (0, 0, 255)),    # Red to blue
    ((0, 255, 0), (0, 0, 255))     # Green to blue
)
```

- **ACCEL_RANGE** allows you to change the range of the accelerometer. You probably want to stick with the default +/-16G range, but you can experiment with smaller ranges by using values like **adafruit_lis3dh.RANGE_8_G**, **adafruit_lis3dh.RANGE_4_G**, or **adafruit_lis3dh.RANGE_2_G**.
- **TAP_THRESHOLD** controls the sensitivity of the spinner. Set to higher values to make the spinner less sensitive.
- **SPINNER_DECAY** controls how quickly the spinner slows down after it starts spinning. Set to a value from 0 to 1.0 where higher values mean the spinner will keep spinning for longer (i.e. less 'friction').
- **COLORS** is a tuple of 2-tuples that specify primary and secondary color combinations.

That's all there is to building the digital fidget spinner with CircuitPython and Circuit Playground express!

MakeCode

There's one final way to build this digital spinner project using the Microsoft MakeCode visual programming tool (https://adafru.it/xCZ). This is a great way to build a digital fidget spinner if you're new to programming and find graphical languages a little easier to understand.

To build the spinner project for MakeCode you'll need to have the **Circuit Playground Express board (https://adafru.it/wpF)**, the older Circuit Playground Classic is not supported.

Next be sure to check out the Circuit Playground Express MakeCode guide (https://adafru.it/xCZ) to learn about MakeCode and how its graphical programming language works. Then load the digital fidget spinner MakeCode project (https://adafru.it/xD0) on your Circuit Playground Express board:

This version of the digital fidget spinner is a little bit simpler than the previous two versions. You only need to shake the Circuit Playground Express to start it spinning its LEDs instead of flicking it from the left or right.

In addition the program won't change colors or animation types from button presses--if you want to change the color adjust the **primary_color** and **secondary_color** variables at the top of the **on start** block in the program.

That's all there is to the MakeCode version of the digital fidget spinner! Shake the board and watch its lights spin around and gradually slow to a stop just like a real fidget spinner!