Make It Shake, Rattle, and Roll: Accelerometer Use

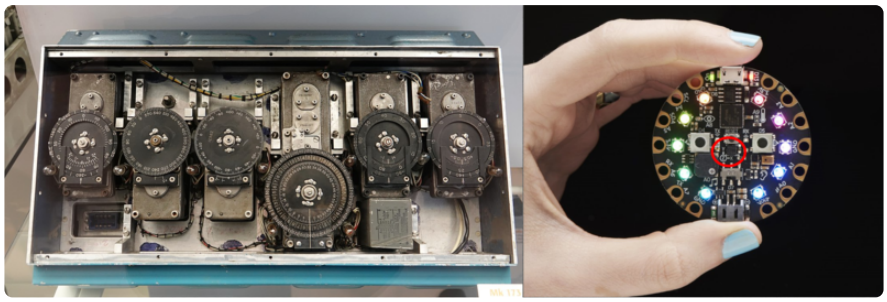
Created by Anne Barela

https://learn.adafruit.com/make-it-shake-rattle-and-roll

Last updated on 2024-03-08 03:08:54 PM EST

Table of Contents

Overview

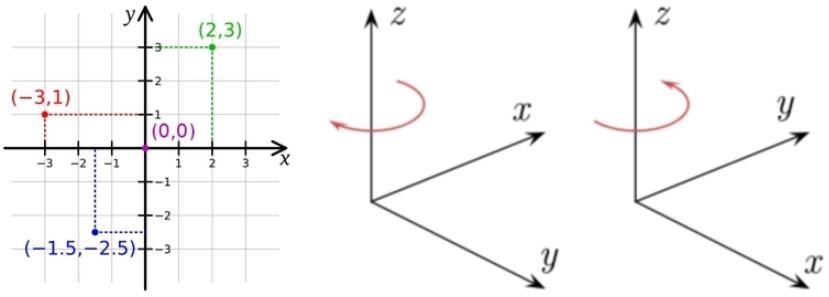


You have a great project idea and it involves action when something moves. How do you turn that idea into a project? Some type of Rube Goldberg machine? Nope - today there are inexpensive, tiny chips called accelerometers which detect motion in three directions. These can make the measurements for you, allowing you can focus on implementing your project!

Adafruit makes several breakout boards using accelerometers for various projects. In addition, the Adafruit Circuit Playground Express (http://adafru.it/3333) has an accelerometer on-board (at the very center) which works great!

This guide will demonstrate the use of an accelerometer in Microsoft MakeCode and CircuitPython using simple projects which will allow you to quickly incorporate movement sensing in your project.

Acceleration & Movement



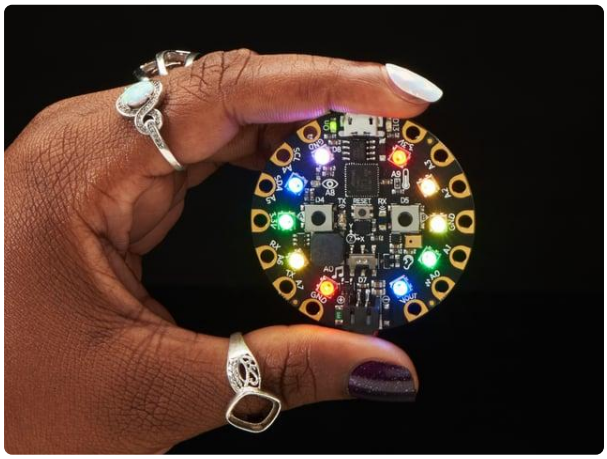
Movement in two dimensions can be thought of as changing the point you are at on a piece of flat paper. Let's make it graph paper so we can count where we might be. If we start at the center, (0, 0), we can move anywhere to the right or left, which we will call the x direction. We can also move up and down (we'll call y). And we can move both up/down and left/right (which would mean changes in both the x and y directions). This is our shake and rattle.

If we now consider that we can lift something off the graph paper (above it or putting it below it), we get one more set of directions, we'll call z. Using three sets of coordinates to point to a place (like 0,0,0) is called 3 dimensions or 3D which we hear alot about. This is the roll.

Now if I move from one place to another, say (0,0,0) to (10,10,10), I have created an acceleration. Wait, what about a velocity, or steady movement? Well, to start at 0 and stop at 10, you have to get going and slow to a stop, both which both are not a steady change in motion. This change in motion which we can relate to a change in where we move something, is called acceleration.

A device which can measure acceleration, changes in motion, is called an accelerometer. Back in the Space Race, an accelerometer was HUGE. Even ten years ago, an accelerometer for one or two directions was rather expensive. Due to cell phone developments, now accelerometer chips are inexpensive and tiny - yay!

Parts



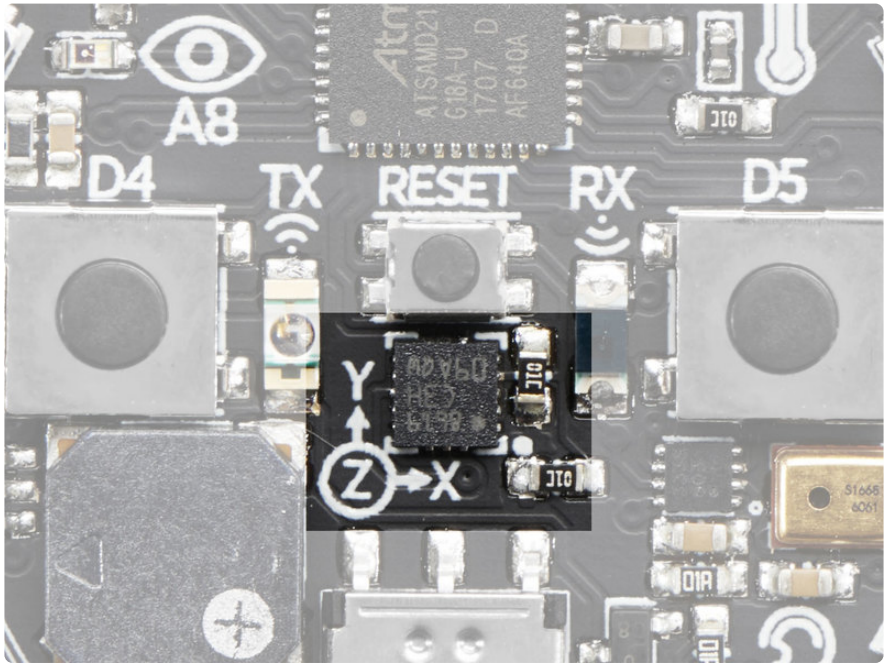
Circuit Playground Express
Circuit Playground Express is the next step towards a perfect introduction to electronics and programming. We've taken the original Circuit Playground Classic and...
https://www.adafruit.com/product/3333

USB cable - USB A to Micro-B
This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...
https://www.adafruit.com/product/592

To Test Using External NeoPixels

Adafruit NeoPixel LED Strip w/ Alligator Clips - 30 LEDs/meter
Adding glowy color to your projects has never been easier: no more soldering or stripping wires, clip 'em on and glow! This Adafruit NeoPixel LED Strip with Alligator...
https://www.adafruit.com/product/3812

The LIS3DH and Other Accelerometers



The Circuit Playground Express has at its center the LIS3DH 3-axis XYZ accelerometer. Some of its features:

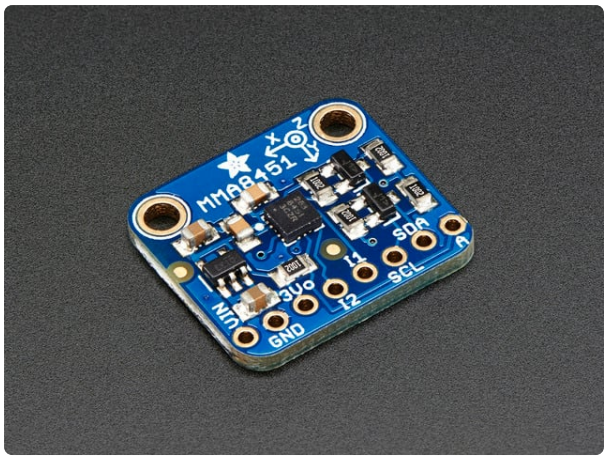
- Three axis sensing, 10-bit precision
- ±2g/±4g/±8g/±16g selectable scaling
- Multiple data rate options 1 Hz to 5Khz
- Tap, Double-tap, orientation & freefall detection

More detail can be found in datasheets and more (https://adafru.it/CuD).

Different accelerometers have separate features. Some have additional bits of precision or additional scaling options but for the price, the LIS3DH provides features most folks will want to implement their projects.

Other Accelerometers

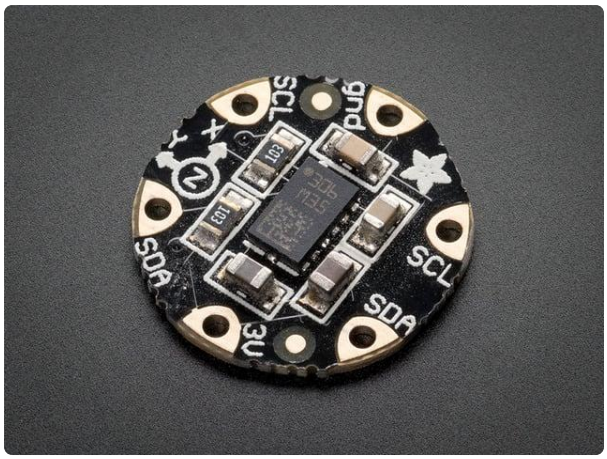
If your project needs differing features, here are some that you may wish to use in your projects other than the LIS3DH depending on differing capabilities:



Adafruit Triple-Axis Accelerometer - ±2/4/8g @ 14-bit - MMA8451

So many accelerometers and so little time! We've expanded our accelerometer selection even more with this high-precision and inexpensive MMA8451 Triple-Axis Accelerometer...

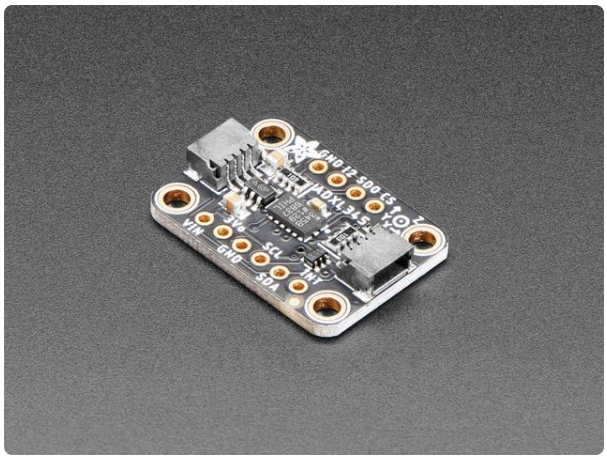
https://www.adafruit.com/product/2019



FLORA Accelerometer/Compass Sensor - LSM303

Add motion and direction sensing to your wearable FLORA project with this high precision 3-axis Accelerometer+Compass sensor. Inside are two sensors, one is a classic 3-axis...

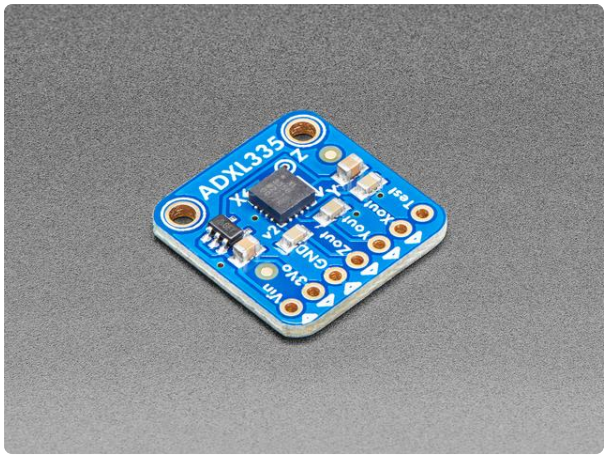
https://www.adafruit.com/product/1247



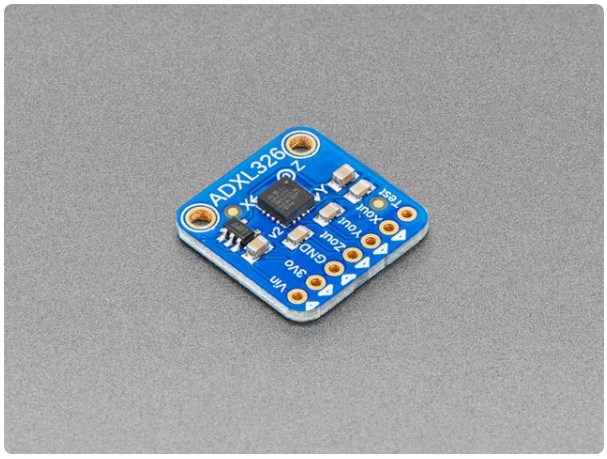
ADXL345 - Triple-Axis Accelerometer (+-2g/4g/8g/16g) w/ I2C/SPI

Filling out our accelerometer offerings, we now have the really lovely digital ADXL345 from Analog Devices, a triple-axis accelerometer with digital I2C and SPI interface breakout. We...

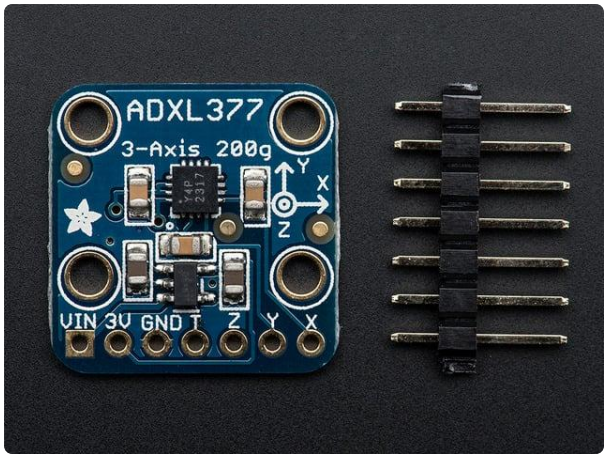
https://www.adafruit.com/product/1231



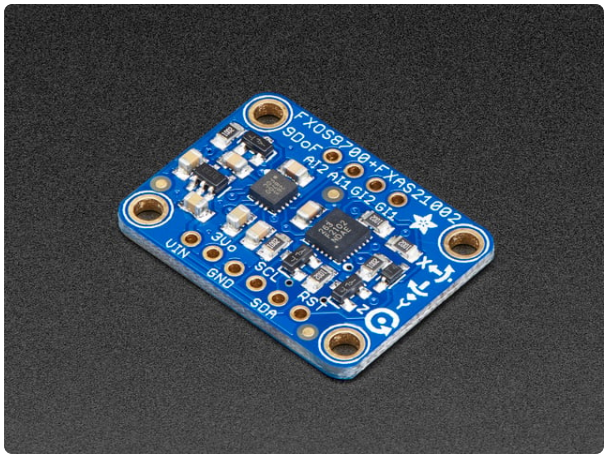
[ADXL335 - 5V ready triple-axis accelerometer (+-3g analog out)]
We've updated our favorite triple-axis accelerometer to now have an on-board 3.3V regulator - making it a perfect choice for interfacing with a 5V microcontroller such as the...
https://www.adafruit.com/product/163



[ADXL326 - 5V ready triple-axis accelerometer (+-16g analog out)]
We've now got a wider range version of favorite triple-axis accelerometer - it even has an on-board 3.3V regulator - making it a perfect choice for interfacing with a 5V...
https://www.adafruit.com/product/1018



[ADXL377 - High-G Triple-Axis Accelerometer (+-200g Analog Out)]
Discontinued - you can grab the ADXL375 - High G Accelerometer (+-200g) with I2C and SPI - STEMMA QT /...
https://www.adafruit.com/product/1413



[Adafruit Precision NXP 9-DOF Breakout Board]
The NXP Precision 9DoF breakout combines two of the best motion sensors we've tested here at Adafruit: The FXOS8700 3-Axis accelerometer and magnetometer, and the...
https://www.adafruit.com/product/3463

Uses for Accelerometers

It's all well and good to hear the mechanics of what an accelerometer is, but how are they useful? What can I do with an accelerometer or what kind of projects might use an accelerometer? Let's go over some uses which may align with what you're looking to do in your projects.

Movement Detection

Any change in the location of the accelerometer chip will be registered as changes in the values it sends out. You can use the fact that the numbers registered have changed to detect a movement, or poll the x, y, and z values and make calculations as to in what direction the motion has taken place and by how much it has moved.

If a project wants to detect any movement or measure a movement, an accelerometer is perfect.

If you are looking for a specific location to where something has moved, an accelerometer will not provide that data, a GPS unit would be a better sensor for exact location data.

Types of projects:

- Sparkle Skirt (https://adafru.it/CuE) / Wearable Activation with Movement (https://adafru.it/CuF)
- Room Door Opening Detection
- Object/Diary Movement
- Electronic Computer Mouse, movement and tapping (https://adafru.it/CuG)
- Detecting Turning Left or Right (https://adafru.it/CuH)
- Making Music through Movement (https://adafru.it/CuI)
- Activation of Something on Start or Stop (https://adafru.it/CuJ)
- Bump Detection, for example in robotics or safety (https://adafru.it/CuK)
- Light Saber Projects (https://adafru.it/CuL)

What's Up and Down

If you need to figure out whether something is oriented the right way with respect to the ground, an accelerometer works fine.

To do so, the accelerometer needs to be oriented so that any movement will be in two directions. With a bit of math, you can calculate "up" and "down" relative to the earth.

> How does this work when you aren't moving the accelerometer? Well don't forget: gravity is a force which makes us accelerate 9.8 meters/s^2 towards the Earth! We can measure that acceleration to see which way is "down"!

Types of Projects:

- Electronic Compass
- "Dialing" a safe combination (https://adafru.it/CuM)
- Inclinometer - how high is a distant object? (https://adafru.it/CuN)

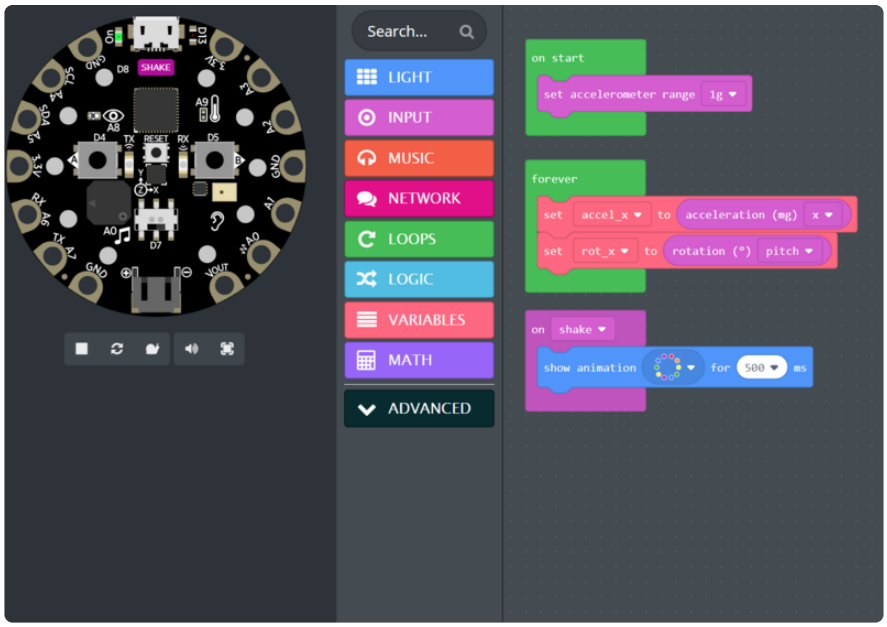
Use in MakeCode

Microsoft MakeCode has excellent support for using the accelerometer on the Circuit Playground Express.

If you are new to using MakeCode and Circuit Playground Express, see this guide (https://adafru.it/wWd) to learn how to get started.

MakeCode Blocks for using the Accelerometer

MakeCode blocks which use the Circuit Playground Express accelerometer are in the **INPUT** code block group.



The `set accelerometer range` block adjusts the range at which the accelerometer will detect. You can think of it as a sensitivity adjustment. If you think the measurements will be small, 1 g should be fine. But for violent bumps, maybe 2, 4, or 8g may be better. Often this block is placed in an `on start` block to set the sensitivity before the main `forever` loop.

The `acceleration` and `rotation` blocks read the accelerometer and provide a value you can use in calculations or decisions. the acceleration will read either x, y, or z so reading all three will take three `set variable` blocks. The value returned is in milli-gs, where one gee is 9.8 m/s2, the pull of the earth. The `acceleration` block can also return a `strength` value which is the acceleration value from all three directions. `rotation` will read either the `pitch` or `roll` values. Pitch is up or down, roll is left or right and is in degrees (0 to 360).

`on shake` is actually a number of readings that can actuate some MakeCode. All the actions you can choose from are:

- `shake`: shake the board
- `logo up`: the logo is facing up
- `logo down`: the logo is facing down
- `screen up`: the screen side is up
- `screen down`: the screen side is down
- `tilt left`: the board is tilted to the left
- `tilt right`: the board is tilted to the right
- `free fall`: the board is falling for a distance
- `3g`: acceleration force of 3 g
- `6g`: acceleration force of 6 g

This is very handy as you don't have to continually read the raw values, compute something, then take action, MakeCode does this for the user if desired.

Examples

The following pages present some examples of using the MakeCode blocks for typical applications.

MakeCode Examples

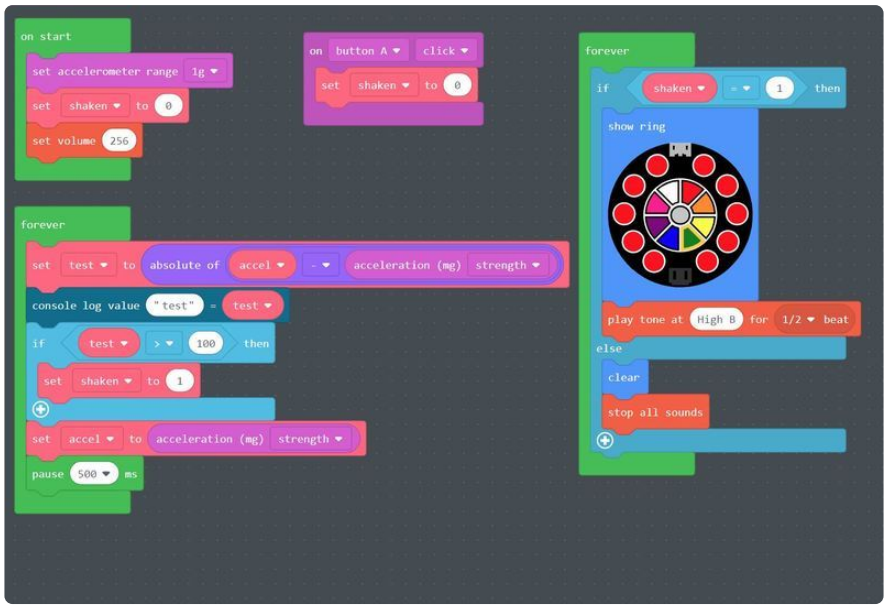
In the **INPUT** block group, select the `on shake` block and put it in the programming area. Click the word **shake** and see all the features that pop up:

You can trigger code to do a great number of things by selecting different versions of this block.

The following code uses the basic shake capability to build an alarm. You can set the sensitivity of the accelerometer in the initial `on start` block with the `set accelerometer range` block.

Detect Movement and Act (Alarm)

This is good for wearables where you want to blink lights and/or make sounds when you move like an alarm. Also this is good for making noise when a Circuit Playground Express is hung on a door or otherwise moved and it alarms.



Open this Example in MakeCode

https://adafru.it/Cva

In `on start`, the accelerometer is set to be very sensitive at 1g, the volume maxed out and a variable `shaken` set to indicate not shaken. The `on shake` block will set the `shaken` variable to `1` to indicate to the `forever` loop that there was a shake. The `forever` loop waits to see if the variable `shaken` is set to 1 by the `on shake` block.

This code also reads the accelerometer `strength` value and if it changes more than a set value (currently `100`), it will also indicate an alarm by setting `shaken` to 1. The `on shake` block was not being "sensitive enough" for an alarm, so some extra code reads the accelerometer `strength`, compares it to a previous reading, and if it has changed by more than a value (currently `100`) it also sets the `shaken = 1` alarm condition.

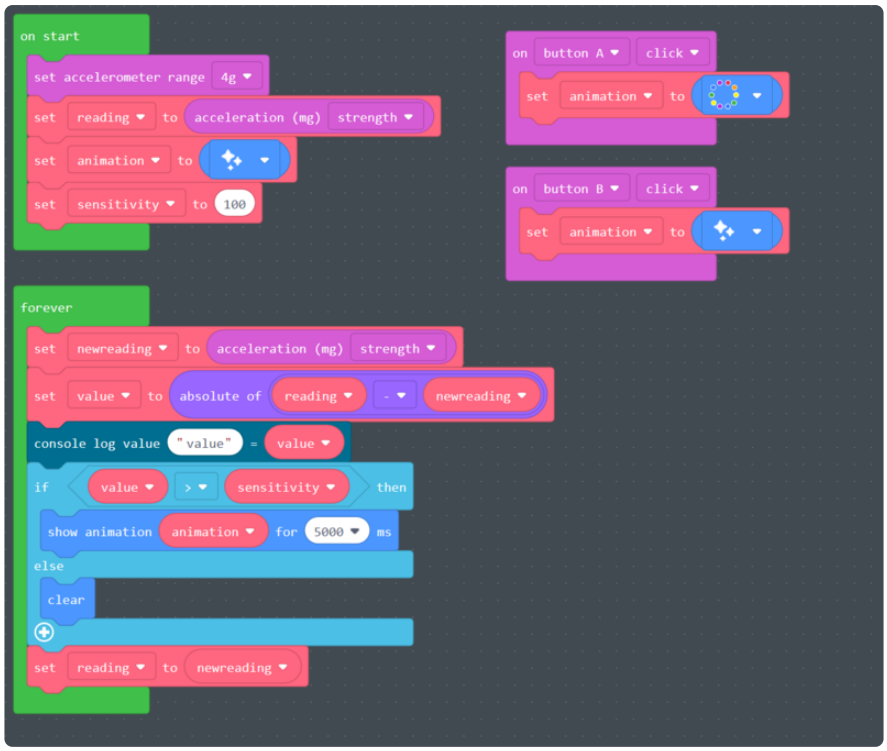
The code gives the programmer control over how sensitive to make it. Change `100` to a value that works for you. If you use MakeCode for Windows 10, you can see what values produce what types of readings to help you set the value of `test`.

If `shaken` has been set to `1`, the device shows all red NeoPixel LEDs and makes a high pitch beep. The way to reset this alarm is if Button A is pressed via `on button A click` (you can make it harder to reset if you like). You might have to hold the Circuit Playground Express real still and press the button a couple times to get it to reset.

More MakeCode

Sparkle Clothing

The code that makes your clothing light up when you move is very similar to the alarm code above. The difference is the movement detection is a bit different and uses twinkling LEDs that turn off when the movement stops.



Open this Example in MakeCode

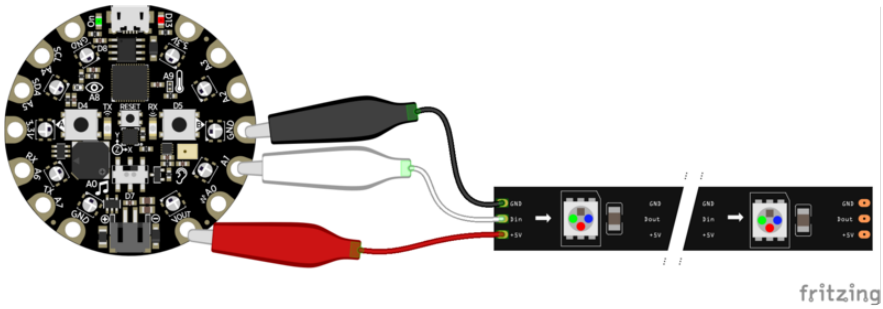
https://adafru.it/CuO

This code uses an `acceleration` range of 4g and uses the `strength` readings of acceleration to read any movement in x, y, or z. If one reading is greater than another reading using some math, the sparkle is "activated" for 5 seconds. The animation can be changed from sparkle to rainbow by pressing button A, sparkle button B. You can adjust the sensitivity by changing the comparison of the variable `sensitivity`. To know what is a good value you can experiment or you can use the Windows 10 version of MakeCode and see the value of the variable `value`.

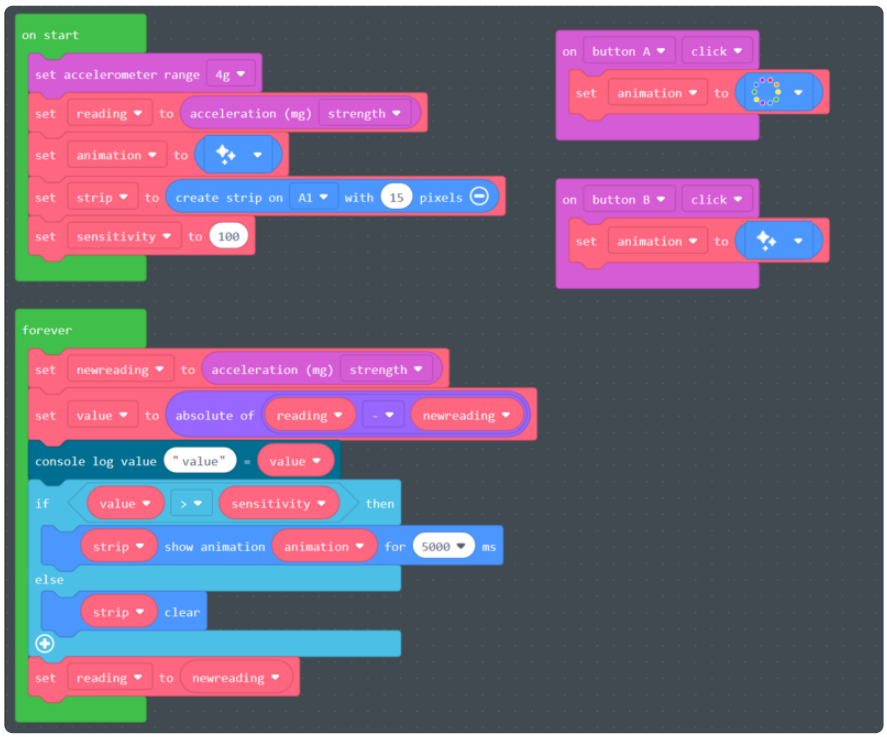
The animation is triggered for 5 seconds (`5000` milliseconds) which may be changed in the `show animation` block. Buttons A and B change the animation from sparkle to rainbow.

External NeoPixels

The example above uses internal NeoPixels on Circuit Playground Express. To animate external NeoPixels, like a strip or individual pixels, you would change the code slightly. Say there are 15 pixels connected to Circuit Playground Express pad A1.



The code would be:



Load this Example in MakeCode

https://adafru.it/CuP

This modified code creates an external strip of 15 pixels (which could be individual pixels or a physical strip or a ring) with the data line connected to Circuit Playground Express pad A1. The animation is shown on the external NeoPixels, not the onboard pixels, and when there is no significant movement the `clear` NeoPixels clears the external pixels.

Buttons A and B change the animation from sparkle to rainbow.

Use in CircuitPython

If you would like to try out CircuitPython, you will find it rather easy to use compared to other programming environments. We suggest you read the Adafruit Welcome to CircuitPython (https://adafru.it/cpy-welcome) Guide to get started.

Adafruit recommends the Mu editor to type your code into and interact with your project. You can find out how to install Mu on mac, PC, and Linux in this guide page (https://adafru.it/ANO) on the Adafruit Learning System.

The advantage to using Mu is it has an editor and added features like direct save to Circuit Playground Express and access to the Express' REPL interactive command line and serial print output. Mu can even plot values for you.

The CPX Library

The `adafruit_circuitplayground` library provides functions for a variety of accelerometer uses including shake detection, tap detection, and reading the accelerometer values.

You can read about the available functions and parameters in this document on read the docs (https://adafru.it/CuQ).

Basic Shake Detection

Here is the basic code for shake detection for the Circuit Playground Express in CircuitPython.

```
import time
from adafruit_circuitplayground import cpx

while True:
    if cpx.shake(shake_threshold=20):
        print("Shake detected!")
        cpx.pixels.fill((150, 0, 0))
        time.sleep(5.0)  # In seconds
    else:
        cpx.pixels.fill((0, 0, 0))
```

This program will make the NeoPixels go red and print a message if a shake is detected.

You can adjust the shake sensitivity easily in the `cpx.shake` detection line. 10 will constantly activate so set it higher than that. The default is 30. The values for this sensitivity are numerically different than those in MakeCode.

Tap Detection

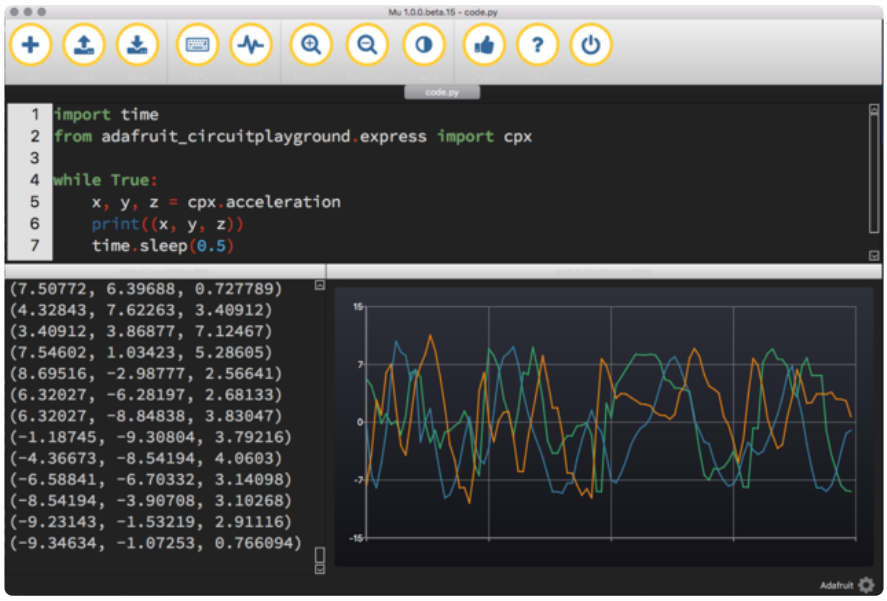
The accelerometer can detect single or double taps. You might use this if you were constructing a mouse or a Star Trek badge, as examples.

```
from adafruit_circuitplayground import cpx

cpx.detect_taps = 2

while True:
    if cpx.tapped:
        print("Tapped!")
```

Change `cpx.detect_taps` to `1` for single tap, `2` for double tap.

Reading Acceleration

You can read the individual x, y, and z values for the accelerometer. The units are in meters/s2 so gravity will pull on the board at 9.8 m/s2 down towards the ground.




```
import time
from adafruit_circuitplayground import cpx

while True:
    x, y, z = cpx.acceleration
    print((x, y, z))
    time.sleep(0.5)
```

This code will print the three values to the serial console and if you are using the Mu editor, you can plot the values. Turn the board in space and see the readings change. See where the earth is pulling?

The video below demonstrates viewing the values.

Note: Their code is a tiny bit different but the numbers output are the exact same numbers the code above is outputting, it is just obtaining them via a lower level library.

You can read more about shake in the guide CircuitPython Made Easy on Circuit Playground Express and Bluefruit (https://adafru.it/19xB).

More CircuitPython

Absolute Acceleration

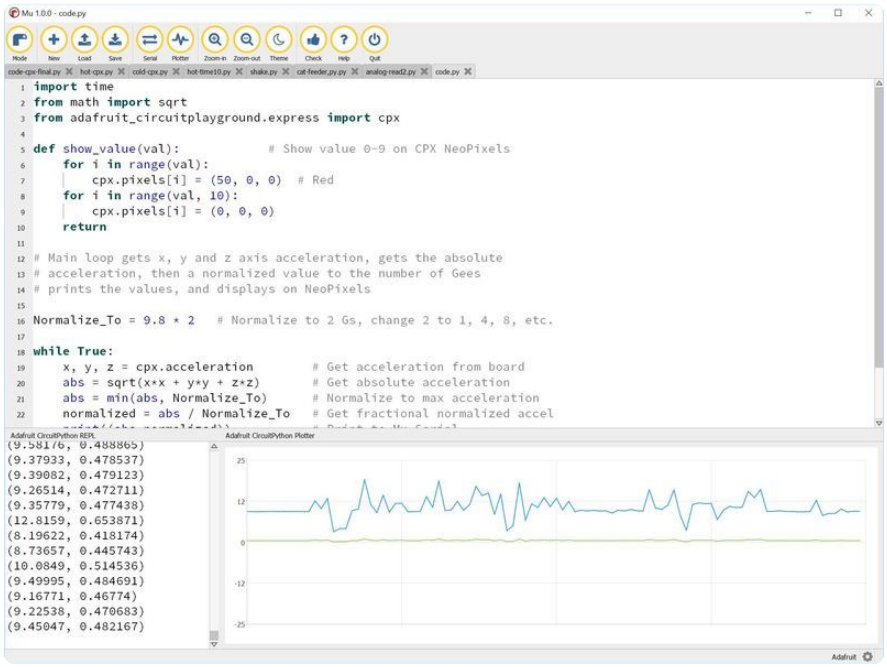
You can get the equivalent of the MakeCode `acceleration strength` block with a tiny bit of math in CircuitPython. The key is to take the square root of the sum of the squares of x, y, and z, similar to the hypotenuse of a triangle but in three dimensions. In vector math, it is getting the magnitude of the acceleration in 3D.

The following CircuitPython code will get the readings from the accelerometer and calculate the absolute acceleration and print it on the serial monitor and plot it in Mu.

```
import time
from math import sqrt
from adafruit_circuitplayground.express import cpx

def show_value(val):            # Show value 0-9 on CPX NeoPixels
    for i in range(val):
        cpx.pixels[i] = (50, 0, 0)  # Red
    for i in range(val, 10):
        cpx.pixels[i] = (0, 0, 0)
    return

# Main loop gets x, y and z axis acceleration, gets the absolute
# acceleration, then a normalized value to the number of Gees
# prints the values, and displays on NeoPixels

Normalize_To = 9.8 * 2   # Normalize to 2 Gs, change 2 to 1, 4, 8, etc.

while True:
    x, y, z = cpx.acceleration        # Get acceleration from board
    abs = sqrt(x*x + y*y + z*z)       # Get absolute acceleration
    abs = min(abs, Normalize_To)      # Normalize to max acceleration
    normalized = abs / Normalize_To   # Get fractional normalized accel
    print((abs,normalized))           # Print to Mu Serial

    show_value(int(normalized * 10))  # Show on NeoPixels on CPX
    time.sleep(0.5)                   # Wait a bit then do again
```

Now shake the accelerometer. Mu will show you the absolute acceleration and the normalized acceleration, here to 2g although you can change this to any value you want, usually 1, 2, 4, 6, 8, 16, etc.

Press the Plotter button and you get a plot. The screen below shows shaking in various directions.



Color Glow Accelerometer

You can use the acceleration values to make a fun light up project with the NeoPixels. There are three acceleration values, and the LEDs have three color values. Let's see what we can do with that!

Copy the following to a file named **code.py** and save to your Circuit Playground Express **CIRCUITPY** drive.

```
    from adafruit_circuitplayground.express import cpx

# Main loop gets x, y and z axis acceleration, prints the values, and turns on
# red, green and blue, at levels related to the x, y and z values.
while True:
    if cpx.switch:
        print("Slide switch off!")
        cpx.pixels.fill((0, 0, 0))
        continue
    else:
        R = 0
        G = 0
        B = 0
        x, y, z = cpx.acceleration
        print((x, y, z))
        if x:
            R = abs(int(x))
        if y:
            G = abs(int(y))
        if z:
            B = abs(int(z))
        cpx.pixels.fill((R, G, B))
```

Move the slide switch to the right if it isn't already. Lights! Now move the board in different directions to see the colors change!

Let's take a look at the code. First we `import` the `adafruit_circuitplayground.express` library as `cpx`.

Inside the `while True:` loop, the program checks to see `if` the switch is to the left. If it is, we `print` `Slide switch off!` and turn off all the LEDs. This creates an "off switch" for the project in case you'd like to leave it sitting around but not have the lights on. `continue` tells the code to keep checking the switch until the state changes, i.e. you move the switch to the right. Once that happens, the code moves on.

Next is the `else` block. First, three variables, R, G and B are created. These will be used to set the colors. They are assigned 0 to start. Then, the code assigns `x, y, z = cpx.acceleration` and prints the values. If you look at the serial output, you'll see how fast the values are scrolling. This is why one typically includes

a `time.sleep()` in the code, to slow those values down to a readable speed. However, this project works best without a `sleep`, so it has been left out.

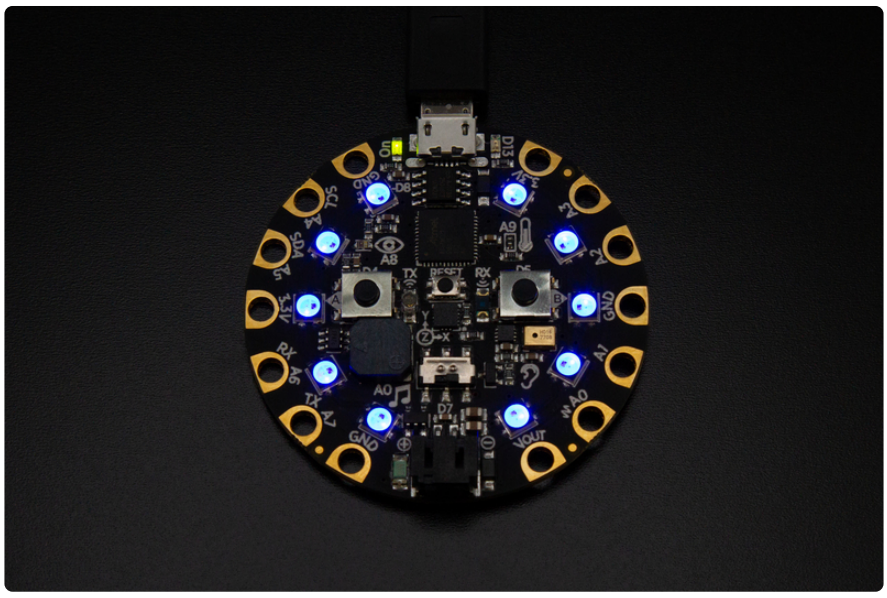
The next section has three parts. Each do the same thing but for a different axis. It is time to take the acceleration values, and use them to represent color values. Begin with `x`.

The value of x is modified a little with `abs(int(x))`. This returns the absolute value of the whole number value of `x`. Absolute values are explained here (https://adafru.it/BnD). Since color values are all whole numbers, use `int(x)` to return only the nearest whole number value of `x`, instead of a long decimal which is often what acceleration returns. Since color values are all positive, take the absolute value of `int(x)` to remove any potential negative numbers from the mix.

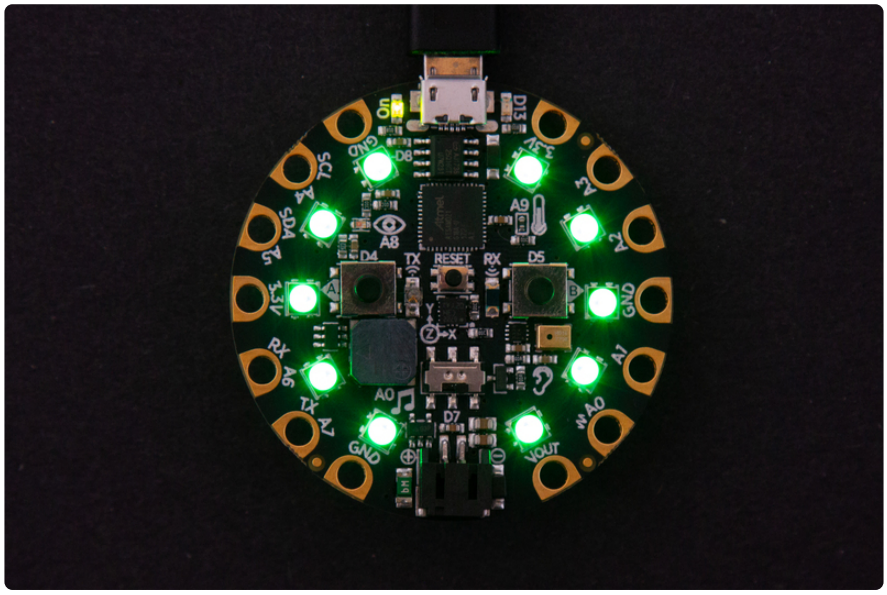
Then take `abs(int(x))` and assign it to `R`. Now we have our `R` value to use for red! Then we do the same thing for `y` and `z`, except `abs(int(y))` is assigned to G and `abs(int(z))` is assigned to B. This gives us three color values!

We then set `cpx.pixels.fill((R, G, B))` to set all the pixels to these color values. As you move the board around, the acceleration values change, and that causes each of our color values to be different. Now, depending on what angle you hold the board, you'll get a different color combination!

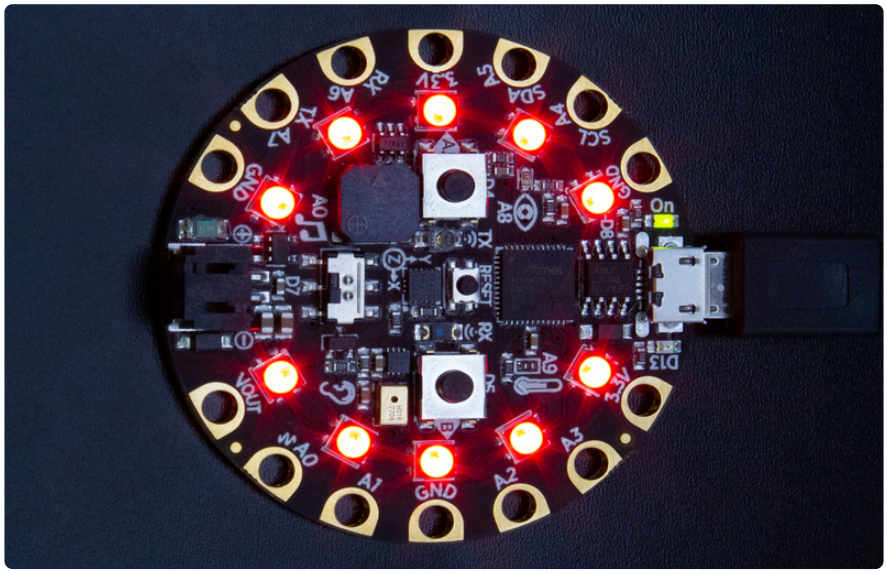
Remember the earlier example, where we explained that if the board is laying flat, the returned values are (0, 0, 9.8). This means, if the board is laying flat, facing up, while this code is running, the color values are `(0, 0, 9.8)`. So, you'll see if it's laying flat on your desk, it's blue!



If you hold it so the USB cable is on the top and pointed downwards, the values are, `(0, 9.8, 0)`, so the LEDs are green.



If you hold it so the USB cable is sideways, pointing left or right, the values are `(9.8, 0, 0)` so the LEDs are red.



As you move the board around at different angles, you'll find every color between!

We also explained that if you shake the board, you'll get back higher values from the accelerometer. This means that the LEDs will be brighter if you shake it. Give it a try!

Accelerate Other Projects

Here are some projects that use acceleration:

- Hacking Ikea Lamps with Circuit Playground Express: CircuitPython Creature Friend (https://adafru.it/Bnt)
- Combo Dial Safe with Circuit Playground Express (https://adafru.it/BnE)