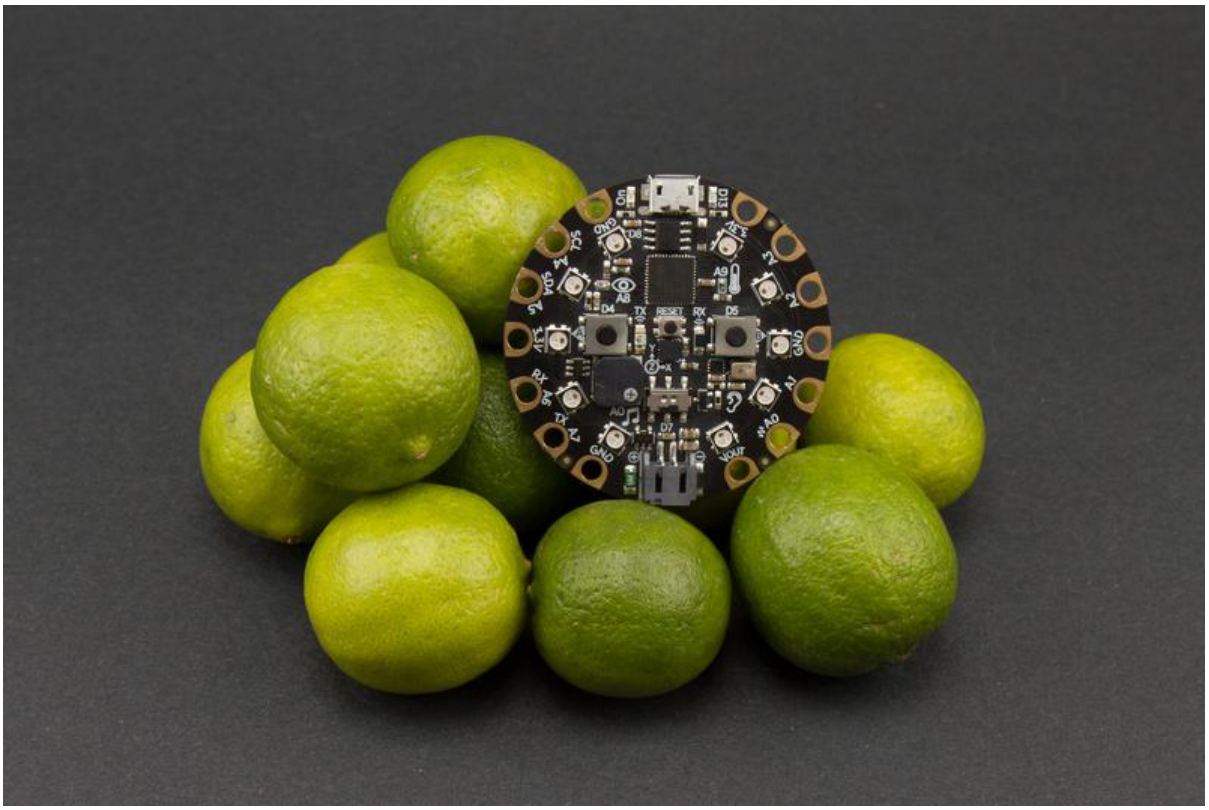
Circuit Playground Express: Piano in the Key of Lime

Created by Kattni Rembor

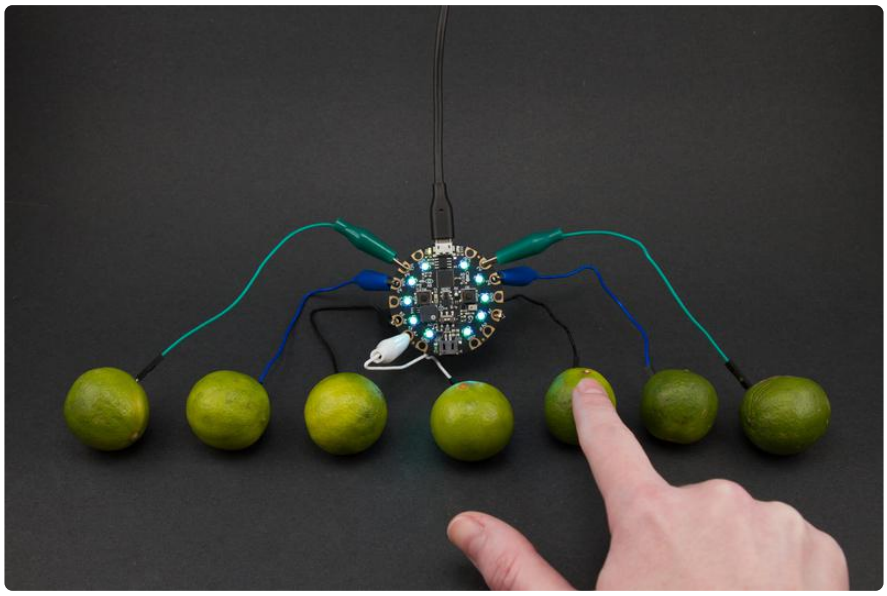


https://learn.adafruit.com/circuit-playground-express-piano-in-the-key-of-lime

Last updated on 2023-08-29 03:25:12 PM EDT

Table of Contents

Overview

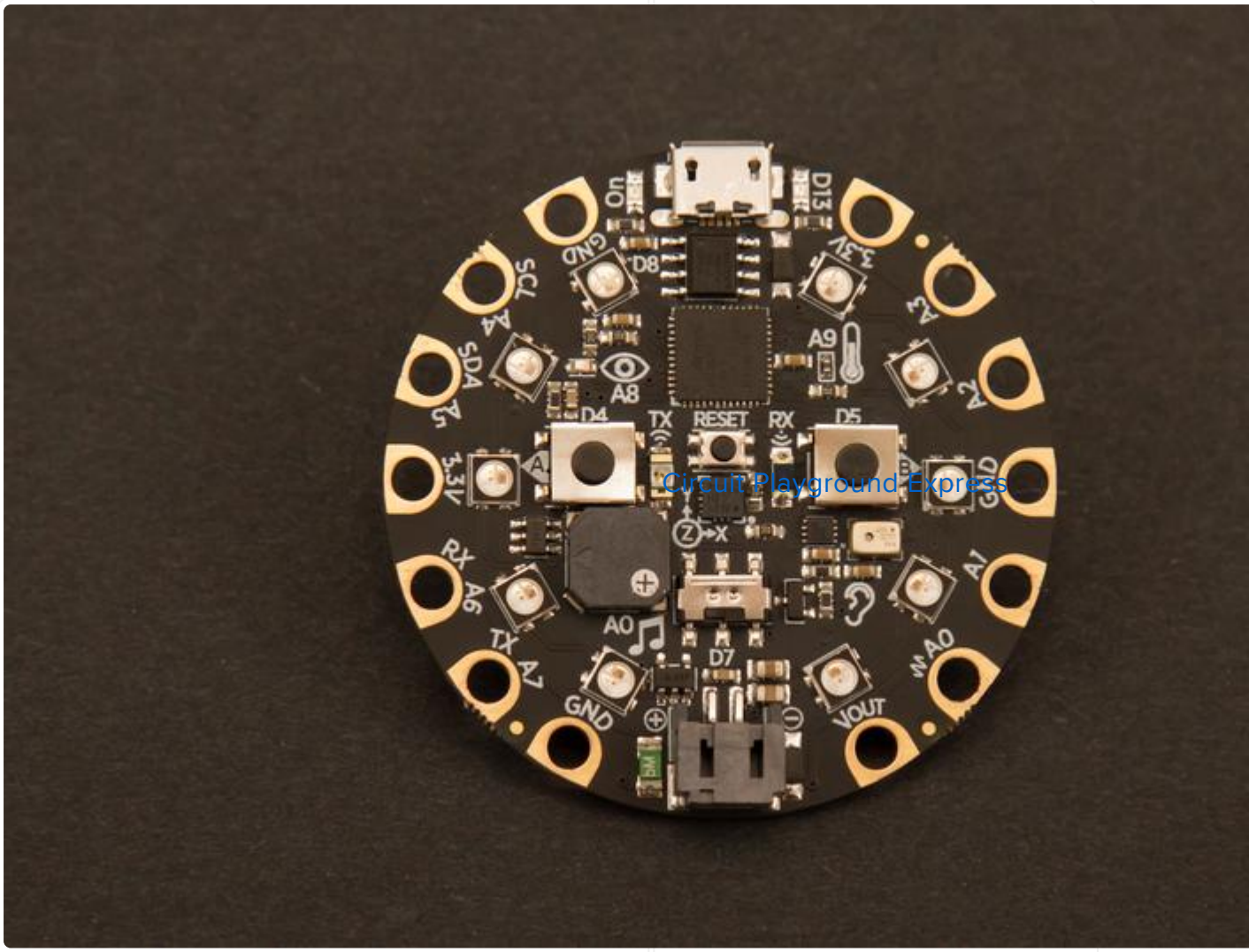


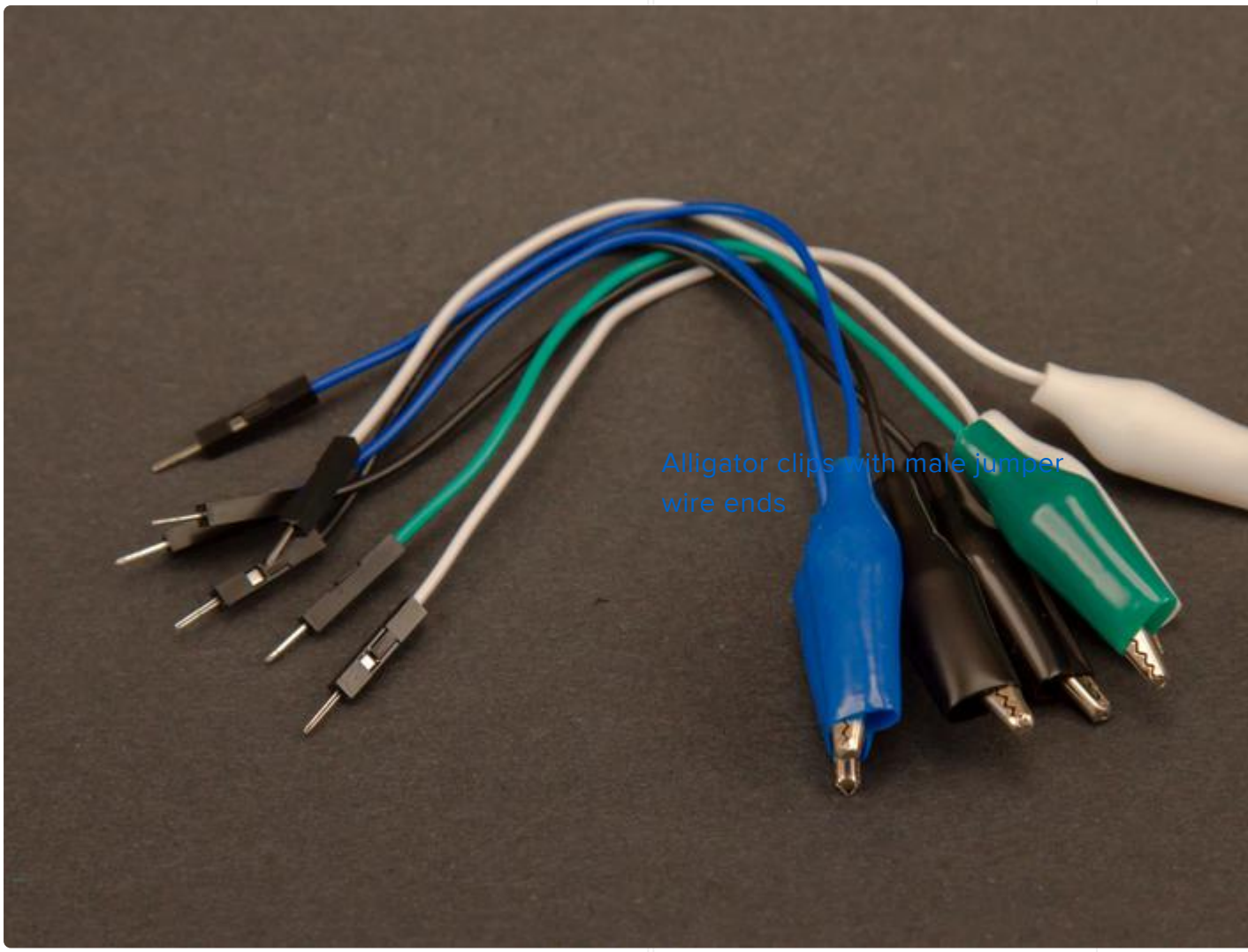
This guide will introduce you to the Circuit Playground Express (CPX for short!) and CircuitPython. The Circuit Playground Express is a fantastic little board crammed full of all sorts of sensors, switches and lights. CircuitPython is an open-source derivative of MicroPython, designed specifically for use with Adafruit microcontroller boards like the CPX.

This project will teach you how to use CircuitPython to utilise the CPX capacitive touch pads, Neopixels, slide switch, and onboard speaker. Then you'll combine them to create a rainbow-lit tone piano!

Required parts

This project will utilise the Circuit Playground Express, connected to a computer via a micro USB cable. Alligator clips will be used to connect to the capacitive touch pads, and jumper wires will be used to extend the cables.


Circuit Playground Express



Alligator clips with male jumper wire ends

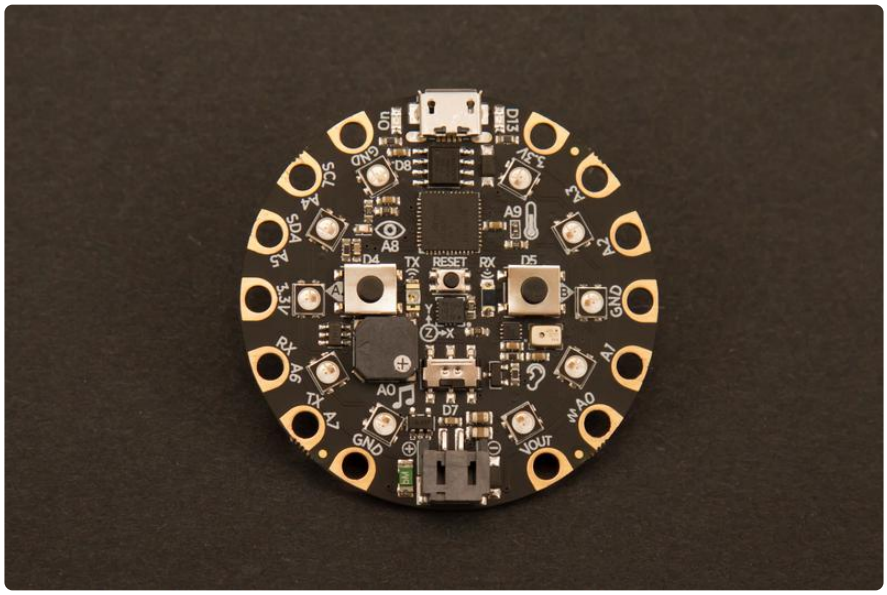
And, the finishing touch: key limes! Key limes are smaller than regular limes, and will make excellent piano keys.

The next sections will introduce you to the CPX and to CircuitPython. Let's get started!

Meet Circuit Playground Express

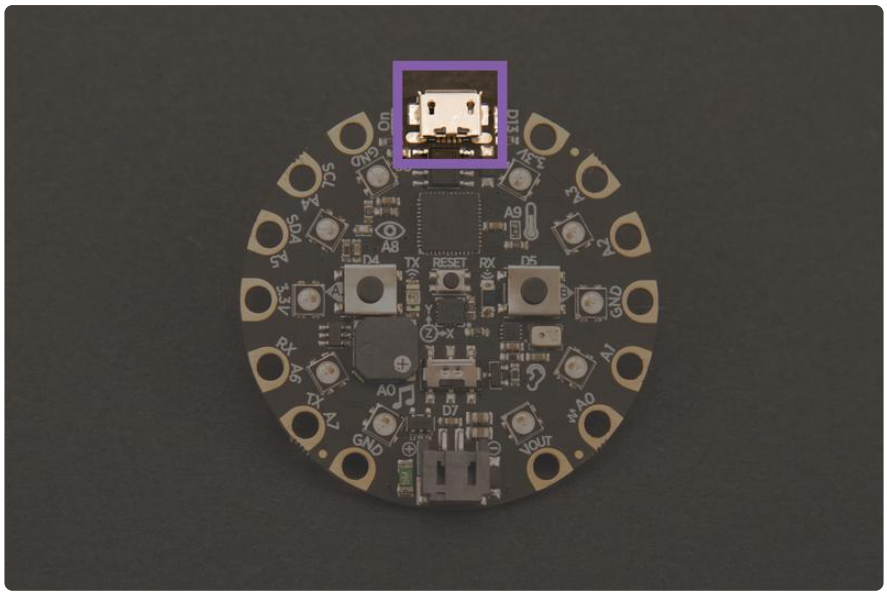
Circuit Playground Express is a microcontroller board that's packed with goodies. Check out this Guided Tour () for all the details. It's designed to work with 3 different programming languages. There's so much to learn with this board!

This project code will be written using CircuitPython. On the CPX, will utilise the micro USB port, capacitive touch sensors, the slide switch, the 10 Neopixels and the onboard speaker.



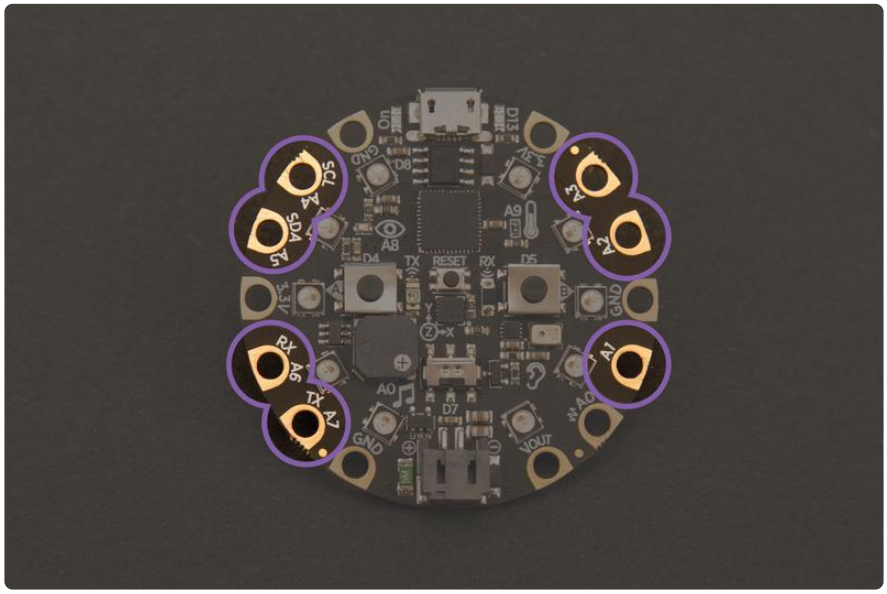
Power using Micro USB

Plug your micro USB data cable into the micro USB port on your CPX. The port is at the top of the board to the right of the power indicator LED.



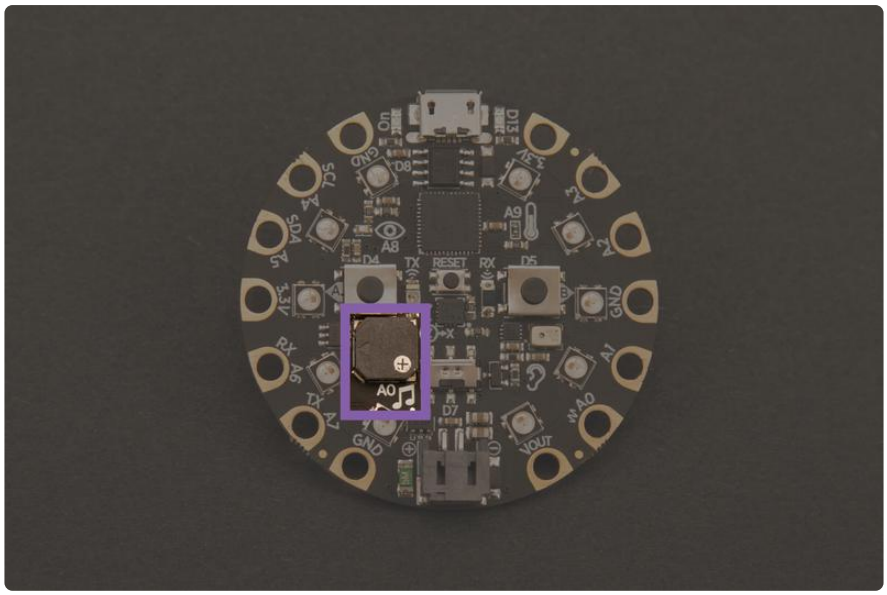
Capacitive Touch Pads

Around the outer edge of the CPX are a fourteen alligator-clip friendly pads. Within the fourteen, are the seven capacitive touch pads, labeled A1-A7. We'll use these to generate tones and lights.



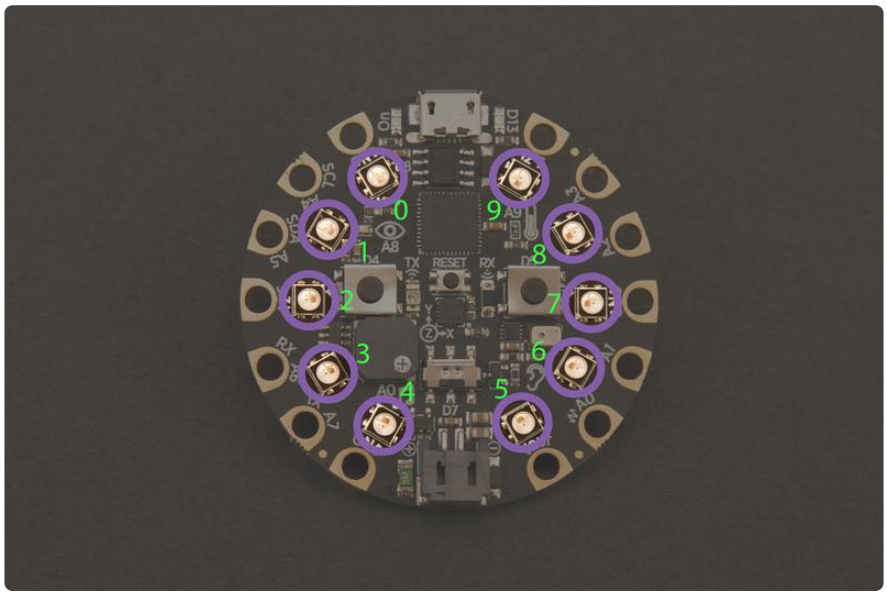
Speaker

The speaker is on the lower left side, above the music notes. We'll use this to generate the tones using the touch pads.



NeoPixels

There are 10 NeoPixels in a ring, just inside the ring of pads. We'll light up these LEDs to go along with our tones.



Next, you'll install CircuitPython on your Circuit Playground Express and get started with CircuitPython.

print("Hello CircuitPython!")

CircuitPython is a version of Python designed to run on tiny computers called microcontrollers. The Circuit Playground Express is a CircuitPython compatible microcontroller. CircuitPython is designed to make getting started with electronics and programming super simple.

CircuitPython looks for a file called `code.py`, and runs the code within automatically. It makes getting started quite straightforward. CircuitPython also allows you to connect to a serial REPL which provides you with a way to see your code working live as you save your files. It also provides you with a prompt where you can enter lines of code and execute them immediately.

In this part of the guide, we're going to cover how to get CircuitPython installed on your Circuit Playground Express and you'll write a little CircuitPython yourself!

Installing CircuitPython on the Circuit Playground Express

Installing CircuitPython is simple. Follow the instructions found on the CircuitPython page of the Adafruit Circuit Playground Express guide (). It includes a few easy steps to get you going with CircuitPython. Once you've finished installing it, head back here to continue.

Congratulations on installing CircuitPython onto your Circuit Playground Express! Way to go!

The Circuit Playground Library

The Circuit Playground library and all libraries it depends on are built into CircuitPython for the Circuit Playground Express. This means you don't need to load any libraries to use it!

The Serial REPL

CircuitPython sends data to the computer connected to the Circuit Playground Express. You can see the output of print statements and any errors you encounter along the way by connecting to the serial REPL.

To get started with the serial REPL, check out the tutorial here: Serial Console (REPL) ( ). There are links in that guide that cover setup on Mac, Linux and Windows.

While this project can be completed without using the REPL, the code is written to give you live feedback through the serial connection. It can be really helpful for troubleshooting problems!

CircuitPython at Its Most Basic

Once you've gotten your serial connection, press `Ctrl+C` then any key to enter the REPL. You should have a prompt that looks like this:

```
>>>
```

This is a Python prompt. Here you can type any python code and have it run.

Every programmer in every programming language starts with a piece of code that says, "Hello, World." We're going to say hello to something else. Next to that prompt, type:

```
print("Hello CircuitPython!")
```

It returns:

```
Hello CircuitPython!
```

Welcome to programming!

Further Reading

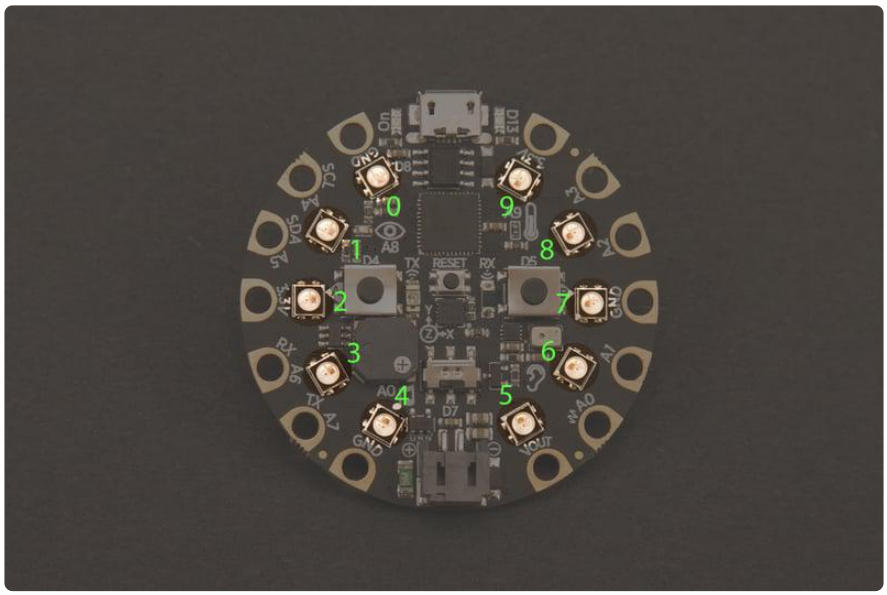
To learn more, take a look at the Introduction to Circuit Playground Express Circuit Python () section. The guide covers all the basics and provides plenty of examples to try and learn from. Check it out!

Light It Up

Circuit Playground Express has 10 Neopixels in a ring. They're numbered 0-9 starting at the top left pixel, next to the micro USB port, and going counter-clockwise around the board. Each pixel can be individually programmed to display any color, or they can be programmed together.

Each NeoPixel LED contains 3 colors: red, green and blue. The colors are collectively referred to as RGB. When working with colored lights, every color is created from

these three being combined at different levels. If only the red is turned on, you'll see red. If red and blue are turned on equally, you'll see purple. If all are turned on equally, you get white. We're going to use this process to add a color of the rainbow to every note on our tone piano.



Let's start by turning on the first Neopixel. We'll turn it blue. Download the following file. Rename it to `code.py`. Copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    cp.pixels[0] = (0, 0, 3)
```

You will be rewarded with a blue pixel on your CPX.

Now let's take a look at the code.

The first line has two parts. The first part is the module. This is different for different types of hardware. As we're using the Circuit Playground Express, we're all good there! The second part of the first line imports the information from the module needed to make the code do what it does. This line will be part of all of the code we use for this project. It's necessary for the board to understand the rest of the code we write.

The second line is telling the CPX to turn on pixel 0, and to make it blue. The colors are written using RGB, which ranges from 0-255. It is the combination of the RGB numbers that change what color is displayed, based on the ratio of the three

numbers. The higher the number, the brighter that individual color is. The three numbers together, separated by commas, are called a tuple. The reason for the two sets of parentheses is that the `pixels()` function is expecting a single piece of information, which is the `(R, G, B)` tuple as a whole..

If you change `[0]` to any number 0 through 9, you can turn on any one of the NeoPixels. If you change the numbers in `((0, 0, 3))` to any number 0-255, you'll change the color and brightness each NeoPixel displays.

You can include more than one NeoPixel by adding another line of code with the number of the desired pixel. The following code lights up the opposite pixel as well. Try editing your `code.py` file to add the additional line of code.

```
from adafruit_circuitplayground.express import cpx

cpx.pixels[0] = ((0, 0, 3))
cpx.pixels[5] = ((3, 0, 0))
```

You can add more pixels or change the colors of the two we've already lit. Have a little fun with it!

Light it up

For this project, we're going to turn all the NeoPixels the same color. We use a different color for each tone. For now, we'll start by turning them all blue.

If you made any changes to your code that you'd like to keep, rename your current `code.py` first. Then, download the following file. Rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    cp.pixels.fill((0, 0, 3))
```

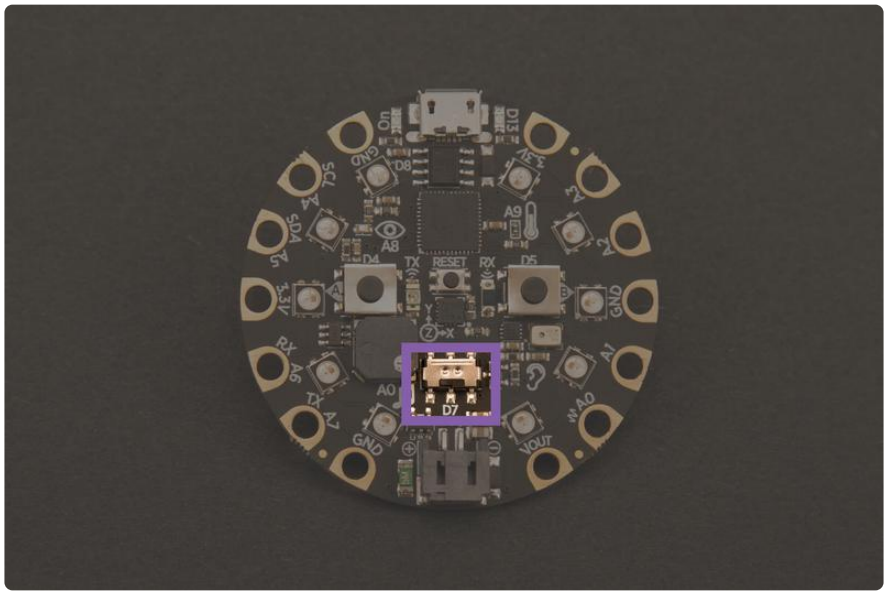
The difference between this code and the first set of code is, instead of specifying an individual NeoPixel by number, we are programming them all at once to display the same color, by using `cp.pixels.fill`.

You can experiment with different colors again by changing the `((0, 0, 3))` to numbers between 0-255.

Next, we'll learn about the slide switch.

Now Slide To The Left

It turns out, when you program the Circuit Playground Express to utilise the capacitive touch for sound, it makes sound pretty much every time you touch it. This is great for the project! However, it can be incredibly frustrating if you're simply trying to move the board or unplug it. To alleviate this issue, we're going to use the slide switch to have the option to mute the tones.



To begin, move the slide switch is to the left. This is the off or mute position, or as the board and code reads it, "`True`".

Rename your current `code.py` if you made any changes you want to keep. Download the following file. Rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    if cp.switch:
        print("Slide switch off!")
    else:
        print("Slide switch on!")
```

We've again imported the module from the library file on the first line. Then we have something new: a `while` statement. `while True:` essentially means, "Forever do:".

`while True:` creates a loop. When there is a loop, the code will forever go through the code inside the loop. All code that is indented under `while True:` is "inside" the loop.

Inside our loop, we have an `if` statement. An `if` statement says, "if this event is happening, do the following." Our code says, if the switch is to the left, or `True`, print "Slide switch off!"

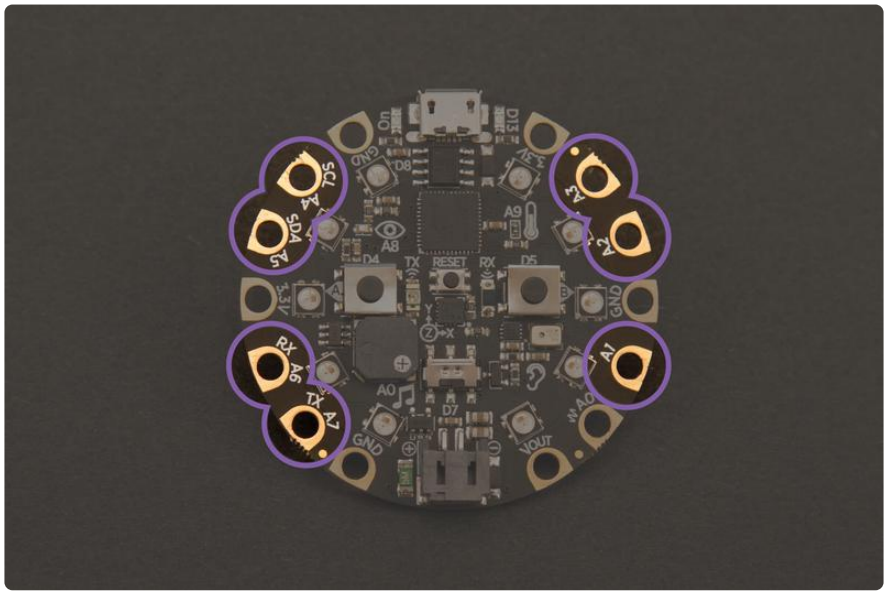
This is followed by an `else` statement. And `else` statement says, "Otherwise, do the following." An `else` typically follows an `if`. Together they say, "If this is happening, do this first thing, otherwise, do the second thing." Our code says, when the switch is to the right, or `False`, print "Slide switch on!"

The `True` and `False` are how the slide switch knows it's location, and do not necessarily reflect the purpose you may assign it.

Next, we're going to learn about the capacitive touch pads and adding sound.

Touch Tone

This section of the guide will bring touch and sound to our project. First, we're going to learn how to use the capacitive touch pads on the Circuit Playground Express. We'll have each one print a response so we know it's working.



We'll start with touch pad A1.

Rename your current `code.py` if you'd like to keep it. Then, download the following file. Rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    if cp.touch_A1:
        print('Touched 1!')
```

Now let's take a look at the code.

In this code, we create a loop. The board is constantly checking to see if you've touched A1, and prints a message for you in the serial REPL when you do. The code basically reads, "If you touch A1, print 'Touched 1!'" That's all there is to it.

The following file includes the rest of the touch pads. Download the following file. Rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    if cp.touch_A1:
        print('Touched 1!')
    elif cp.touch_A2:
        print('Touched 2!')
    elif cp.touch_A3:
        print('Touched 3!')
    elif cp.touch_A4:
        print('Touched 4!')
    elif cp.touch_A5:
        print('Touched 5!')
    elif cp.touch_A6:
        print('Touched 6!')
    elif cp.touch_A7:
        print('Touched 7!')
```

Now you can touch any of the capacitive touch pads and the serial REPL will let you know which one you touch. Give it a try!

Let's look at the difference in the code.

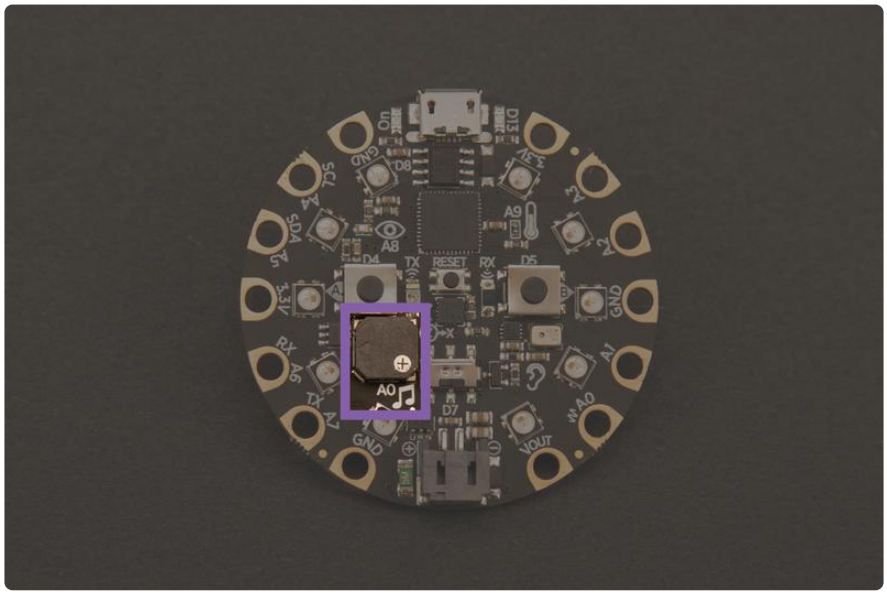
```
while True:
    if cp.touch_A1:
        print('touched 1!')
    elif cp.touch_A2:
        print('touched 2!')
```

So far we've used `while` loops, `if`, and `else` statements. This code includes an `elif` statement. `elif` is combination of `else` and `if`, which essentially means, "Otherwise, if."

There are 7 touch pads. The board needs to know which one you're touching, and provide a response based on which pad you're touching. The code says, "If you're touching A1, I will print '`Touched 1!`', otherwise, if you're touching A2, I will print '`Touched 2!`', otherwise, if you're touching A3, I will print '`Touched 3!`'" and so on for each touch pad. We cannot use `else` in this piece of code because this isn't simply an "on/off" situation. It's more granular with 7 possible states, one for each touch pad.

Make Some Noise

Now we'll learn how to make tones using the onboard speaker.



There are two ways to make tones using the onboard speaker. One is `play_tone`, which plays a given tone for a given duration. The other is `start_tone` and `stop_tone`, which require you to provide a trigger, such as pressing a button (or touching a touch pad!), to play a tone for the duration of the trigger event.

We'll start by making a single tone. Download the file. Rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp
```

```
cp.play_tone(440, 1)
```

In the first example, we use `play_tone`, which requires you to provide a frequency (in Hz) and a duration (in seconds). We've provided 440 Hz (which is Middle A) and 1 second.

In this case, the tone plays as soon as the code runs, which, since there is no other code included, occurs as soon as the board starts up. This code doesn't repeat the tone - it plays it once. You can save your `code.py` again, or reload the serial REPL to make it play the tone again.

You can change both the frequency and duration to change the pitch of the tone and the length for which it plays. If you're looking for specific pitches, check out an online tone generator. Have have fun with it!

Start Tone, Stop Tone

For our tone piano, we'll be using `start_tone` and `stop_tone`. For this example, we'll use touch pads A1 and A2 on your Circuit Playground Express.

Remember, if you made any changes to your code that you'd like to keep, rename your current `code.py`. Download the following file, rename it to `code.py` and copy it to your CPX.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    if cp.touch_A1:
        cp.start_tone(262)
    elif cp.touch_A2:
        cp.start_tone(294)
    else:
        cp.stop_tone()
```

This second example uses `start_tone`, which requires you to provide a frequency (in Hz), and `stop_tone` which does exactly that, stops the tone.

If you touch A1, you'll hear a tone until you stop touching A1. If you touch A2, you'll hear a tone until you stop touching A2. We've again used `while`, `if`, and `elif` and, this time, we included `else`. This code essentially means, "As long as you're touching A1, play the given tone, otherwise stop playing. Otherwise as long as you're

touching A2, play the given tone, otherwise stop playing." The `else` statement applies to both the `if` and the `elif` separately.

This is the start of a touch tone piano! We're almost there. In the next section, we'll put together everything we've learned into the final project!

Piano in the Key of Lime

Now we'll take everything we learned and put it together!

Be sure to save your current code.py if you've changed anything you'd like to keep. Download the following file. Rename it to code.py and save it to your Circuit Playground Express.

```
# SPDX-FileCopyrightText: 2017 Kattni Rembor for Adafruit Industries
#
# SPDX-License-Identifier: MIT

from adafruit_circuitplayground import cp

while True:
    if cp.switch:
        print("Slide switch off!")
        cp.pixels.fill((0, 0, 0))
        cp.stop_tone()
        continue
    if cp.touch_A4:
        print('Touched A4!')
        cp.pixels.fill((15, 0, 0))
        cp.start_tone(262)
    elif cp.touch_A5:
        print('Touched A5!')
        cp.pixels.fill((15, 5, 0))
        cp.start_tone(294)
    elif cp.touch_A6:
        print('Touched A6!')
        cp.pixels.fill((15, 15, 0))
        cp.start_tone(330)
    elif cp.touch_A7:
        print('Touched A7!')
        cp.pixels.fill((0, 15, 0))
        cp.start_tone(349)
    elif cp.touch_A1:
        print('Touched A1!')
        cp.pixels.fill((0, 15, 15))
        cp.start_tone(392)
    elif cp.touch_A2 and not cp.touch_A3:
        print('Touched A2!')
        cp.pixels.fill((0, 0, 15))
        cp.start_tone(440)
    elif cp.touch_A3 and not cp.touch_A2:
        print('Touched A3!')
        cp.pixels.fill((5, 0, 15))
        cp.start_tone(494)
    elif cp.touch_A2 and cp.touch_A3:
        print('Touched "8"!')
        cp.pixels.fill((15, 0, 15))
        cp.start_tone(523)
    else:
```

```
        cp.pixels.fill((0, 0, 0))
        cp.stop_tone()
```

Now, we're going to break down the code.

The first thing we do is setup the slide switch to be able to turn the project on and off.

```
while True:
    if cp.switch:
        print("Slide switch off!")
        cp.pixels.fill((0, 0, 0))
        cp.stop_tone()
        continue
```

This is the beginning code. All of the code in this project is inside this `while` loop. The first thing the code does inside the loop is check whether the slide switch is to the left. For the purposes of this project, left means "Off." If it is off, the code prints, "`Slide switch off!`", turns off the LEDs, and stops playing any sound. Then, `continue` ensures that the code doesn't stop here, and instead continues on to the next sections.

Next, we're going to code each touch pad to play a different tone and display a different color on the NeoPixels. We'll leave the print statements in so we have some feedback in the REPL.

A piano plays low to high from left to right. To allow our "keys" to be in the correct configuration using the alligator clip jumper wires to spread them out, we'll need to do things a bit out of order. We'll code the touch pads in the same order that we'll place the limes, starting on the left. We'll start with A4:

```
if cpx.touch_A4:
        print('Touched A4!')
        cp.pixels.fill((15, 0, 0))
        cp.start_tone(262)
```

If you touch A4, the code prints "`Touched A4!`" to the REPL, lights up the NeoPixels red, and plays a tone at 262 Hz (which is Middle C). This is the first "key" in our touch piano! A5, A6, A7 and A1 are essentially the same. For each one, we change the message, color and tone. These will use `elif` statements instead of `if`.

There are 7 touch pads. We're going to include 8 tones to create a full scale. To simulate the 8th touch pad, you will touch both A2 and A3 at the same time. Therefore, we have to write the code to tell the difference between A2, A3 and both at the same time.

Let's take a look:

```
elif cp.touch_A2 and not cpx.touch_A3:
        print('Touched A2!')
        cp.pixels.fill((0, 0, 15))
        cp.start_tone(440)
    elif cp.touch_A3 and not cpx.touch_A2:
        print('Touched A3!')
        cp.pixels.fill((5, 0, 15))
        cp.start_tone(494)
```

If you touch A2 and not A3, the code prints, "`Touched A2!`", displays the given color and plays the given tone. If you touch A3 and not A2, the code prints, "`Touched A3!`", displays the given color and plays the given tone.

The simulated touch pad looks like this:

```
elif cp.touch_A2 and cp.touch_A3:
        print('Touched "8"!')
        cp.pixels.fill((15, 0, 15))
        cp.start_tone(523)
```

If you are touching A2 and A3, the code prints "`Touched "8"!`", displays the final color and plays the final tone.

The last piece of the code says if you're not touching anything, turn off the lights and stop playing the tone.

```
else:
        cp.stop_tone()
        cp.pixels.fill((0, 0, 0))
```

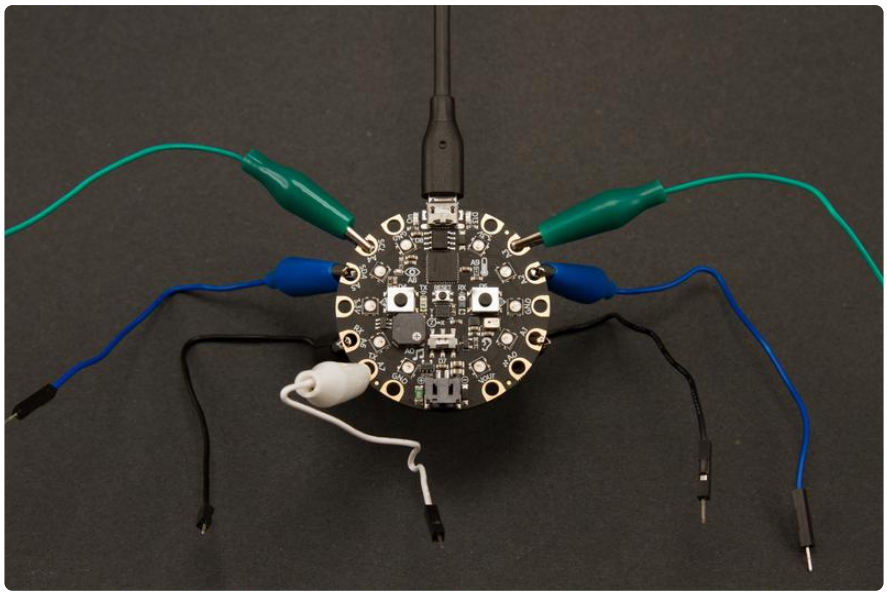
Now, if you touch the pads, you'll hear all the tones and see all the colors we assigned!

Making the Key of Lime

Now we'll add our fruit keys!

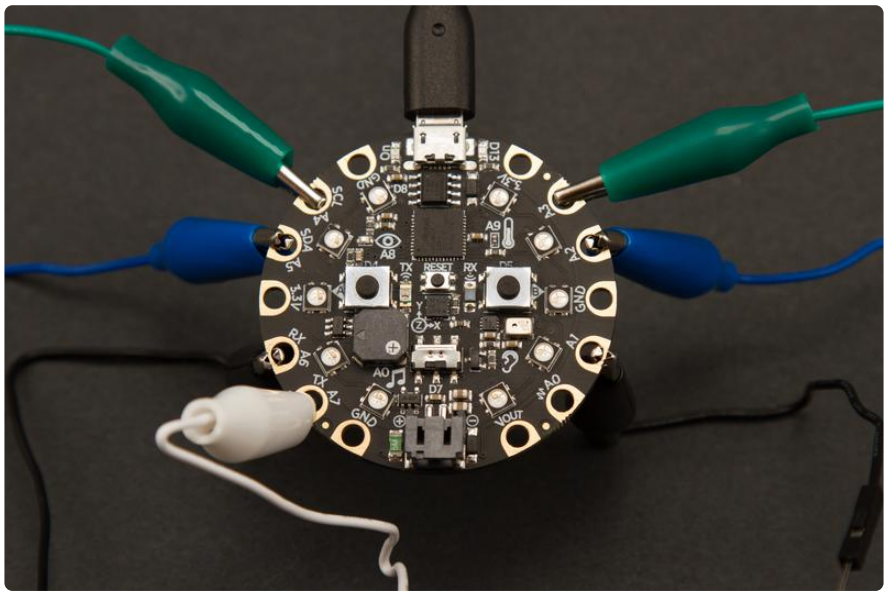
Power off your Circuit Playground Express. Attach one alligator clip jumper wire to each touch pad.

The capacitive touch pads calibrate on startup, so if you leave your CPX powered on, you may need to hit the reset button after attaching the fruit.



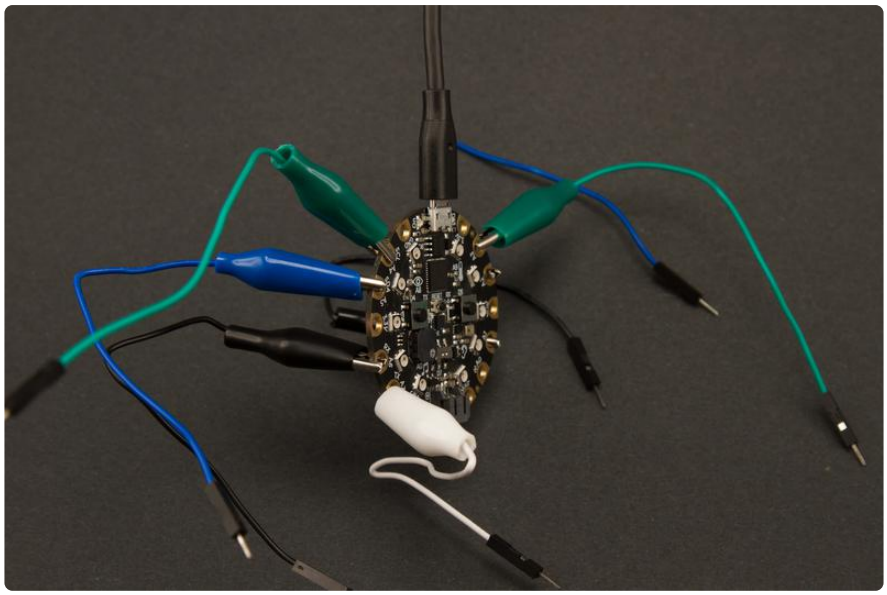
You'll want to arrange them so they don't touch. You can slightly crease the wires on your clips to persuade them to stay in a particular place. If the wires are touching, they can cause the attached pads to act like they're being touched.

This arrangement works consistently:

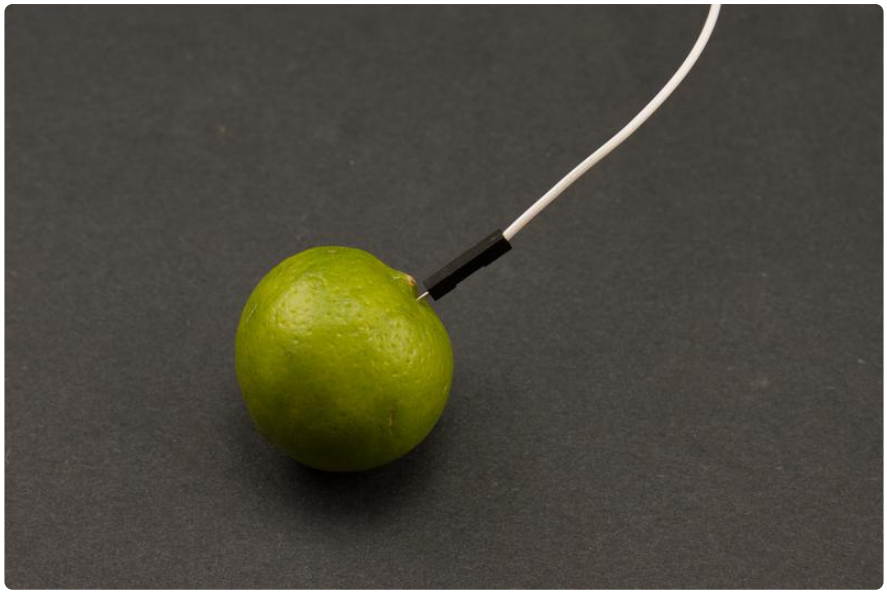


The clips are attached in a way that allows them to fan out, and avoid interference from the other clips.

Note the clip attached to A7 is creased twice to shorten it up.

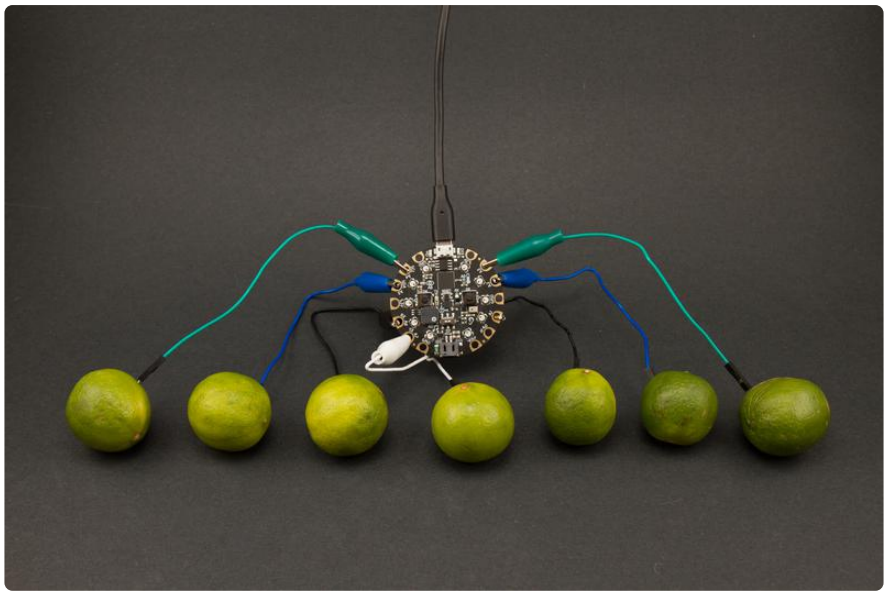


Now you'll attach the fruits. With key limes, it's easiest to simply choose a location and puncture the skin with the male jumper wire end.



Now arrange the the limes in a line, but not too close together. As with the wires, placing the limes too close together can cause interference.

Plug in your Circuit Playground Express. You may need to press the reset button after initially powering it on if it's plugged into a battery.



Now it's time to play! This video shows a song being played on the Piano in the Key of Lime. Have fun playing!