

---

# **Adafruit CircuitPlayground Library Documentation**

***Release 1.0***

**Scott Shawcroft**

**Sep 16, 2019**



---

## Contents

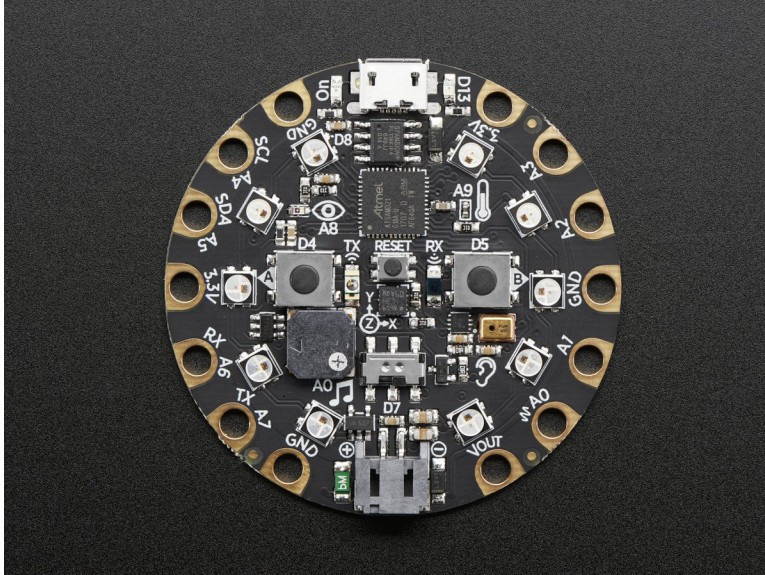
---

<b>1</b>	<b>Installation</b>	<b>3</b>
<b>2</b>	<b>Usage Example</b>	<b>5</b>
<b>3</b>	<b>Contributing</b>	<b>7</b>
<b>4</b>	<b>Building locally</b>	<b>9</b>
4.1	Sphinx documentation . . . . .	9
<b>5</b>	<b>Table of Contents</b>	<b>11</b>
5.1	Simple test . . . . .	11
5.2	adafruit_circuitplayground.express . . . . .	24
<b>6</b>	<b>Indices and tables</b>	<b>43</b>
	<b>Python Module Index</b>	<b>45</b>
	<b>Index</b>	<b>47</b>





This high level library provides objects that represent CircuitPlayground hardware.





# CHAPTER 1

---

## Installation

---

This driver depends on many other libraries! Please install it by downloading [the Adafruit library and driver bundle](#).



## CHAPTER 2

---

### Usage Example

---

Using it is super simple. Simply import the `cpx` variable from the module and then use it.

```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.button_a:
        print("Temperature:", cpx.temperature)
    cpx.red_led = cpx.button_b
```



## CHAPTER 3

---

### Contributing

---

Contributions are welcome! Please read our [Code of Conduct](#) before contributing to help this project stay welcoming.





## CHAPTER 4

---

### Building locally

---

To build this library locally you'll need to install the `circuitpython-build-tools` package.

```
python3 -m venv .env
source .env/bin/activate
pip install circuitpython-build-tools
```

Once installed, make sure you are in the virtual environment:

```
source .env/bin/activate
```

Then run the build:

```
circuitpython-build-bundles --filename_prefix adafruit-circuitpython-
↳circuitplayground --library_location .
```

### 4.1 Sphinx documentation

Sphinx is used to build the documentation based on rST files and comments in the code. First, install dependencies (feel free to reuse the virtual environment from above):

```
python3 -m venv .env
source .env/bin/activate
pip install Sphinx sphinx-rtd-theme
```

Now, once you have the virtual environment activated:

```
cd docs
sphinx-build -E -W -b html . _build/html
```

This will output the documentation to `docs/_build/html`. Open the `index.html` in your browser to view them. It will also (due to `-W`) error out on any warning like Travis will. This is a good way to locally verify it will pass.



## 5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/circuitplayground\_acceleration.py

```
1 import time
2 from adafruit_circuitplayground.express import cpx
3
4 while True:
5     x, y, z = cpx.acceleration
6     print(x, y, z)
7
8     time.sleep(0.1)
```

Listing 2: examples/circuitplayground\_pixels\_simpletest.py

```
1 # CircuitPython demo - NeoPixel
2
3 import time
4 from adafruit_circuitplayground.express import cpx
5
6 # The number of pixels in the strip
7 numpix = 10
8
9
10 def wheel(pos):
11     # Input a value 0 to 255 to get a color value.
12     # The colours are a transition r - g - b - back to r.
13     if (pos < 0) or (pos > 255):
14         return (0, 0, 0)
15     if pos < 85:
16         return (int(pos * 3), int(255 - (pos*3)), 0)
```

(continues on next page)

(continued from previous page)

```

17     if pos < 170:
18         pos -= 85
19         return (int(255 - pos*3), 0, int(pos*3))
20     pos -= 170
21     return (0, int(pos*3), int(255 - pos*3))
22
23
24 def rainbow_cycle(wait):
25     for j in range(255):
26         for i in range(cpx.pixels.n):
27             idx = int((i * 256 / len(cpx.pixels)) + j)
28             cpx.pixels[i] = wheel(idx & 255)
29             cpx.pixels.show()
30             time.sleep(wait)
31
32
33 cpx.pixels.auto_write = False
34 cpx.pixels.brightness = 0.3
35 while True:
36     rainbow_cycle(0.001)    # rainbowcycle with 1ms delay per step

```

Listing 3: examples/circuitplayground\_shake.py

```

1 from adafruit_circuitplayground.express import cpx
2
3 while True:
4     if cpx.shake(shake_threshold=20):
5         print("Shake detected!")

```

Listing 4: examples/circuitplayground\_tapdetect\_single\_double.py

```

1 from adafruit_circuitplayground.express import cpx
2
3 # Set to check for single-taps.
4 cpx.detect_taps = 1
5 tap_count = 0
6
7 # We're looking for 2 single-taps before moving on.
8 while tap_count < 2:
9     if cpx.tapped:
10         tap_count += 1
11 print("Reached 2 single-taps!")
12
13 # Now switch to checking for double-taps
14 tap_count = 0
15 cpx.detect_taps = 2
16
17 # We're looking for 2 double-taps before moving on.
18 while tap_count < 2:
19     if cpx.tapped:
20         tap_count += 1
21 print("Reached 2 double-taps!")
22 print("Done.")

```

Listing 5: examples/circuitplayground\_tapdetect.py

```

1 from adafruit_circuitplayground.express import cpx
2
3 cpx.detect_taps = 1
4
5 while True:
6     if cpx.tapped:
7         print("Single tap detected!")

```

Listing 6: examples/circuitplayground\_tone.py

```

1 from adafruit_circuitplayground.express import cpx
2
3 while True:
4     if cpx.button_a:
5         cpx.start_tone(262)
6     elif cpx.button_b:
7         cpx.start_tone(294)
8     else:
9         cpx.stop_tone()

```

Listing 7: examples/circuitplayground\_touched.py

```

1 from adafruit_circuitplayground.express import cpx
2
3 while True:
4     if cpx.touch_A1:
5         print('Touched pad A1')
6     if cpx.touch_A2:
7         print('Touched pad A2')
8     if cpx.touch_A3:
9         print('Touched pad A3')
10    if cpx.touch_A4:
11        print('Touched pad A4')
12    if cpx.touch_A5:
13        print('Touched pad A5')
14    if cpx.touch_A6:
15        print('Touched pad A6')
16    if cpx.touch_A7:
17        print('Touched pad A7')

```

Listing 8: examples/circuitplayground\_acceleration\_neopixels.py

```

1 """If the switch is to the right, it will appear that nothing is happening. Move the_
2 ↪switch to the
3 left to see the NeoPixels light up in colors related to the accelerometer! The CPX_
4 ↪has an
5 accelerometer in the center that returns (x, y, z) acceleration values. This program_
6 ↪uses those
7 values to light up the NeoPixels based on those acceleration values."""
8 from adafruit_circuitplayground.express import cpx
9
10 # Main loop gets x, y and z axis acceleration, prints the values, and turns on
11 # red, green and blue, at levels related to the x, y and z values.
12 while True:

```

(continues on next page)

(continued from previous page)

```

10     if not cpx.switch:
11         # If the switch is to the right, it returns False!
12         print("Slide switch off!")
13         cpx.pixels.fill((0, 0, 0))
14         continue
15     else:
16         R = 0
17         G = 0
18         B = 0
19         x, y, z = cpx.acceleration
20         print((x, y, z))
21         cpx.pixels.fill(((R + abs(int(x))), (G + abs(int(y))), (B + abs(int(z)))))

```

Listing 9: examples/circuitplayground\_button\_a.py

```

1  """This example turns on the little red LED when button A is pressed."""
2  from adafruit_circuitplayground.express import cpx
3
4  while True:
5      if cpx.button_a:
6          print("Button A pressed!")
7          cpx.red_led = True

```

Listing 10: examples/circuitplayground\_button\_b.py

```

1  """This example turns the little red LED on only while button B is currently being_
↪pressed."""
2  from adafruit_circuitplayground.express import cpx
3
4  # This code is written to be readable versus being Pylint compliant.
5  # pylint: disable=simplifiable-if-statement
6
7  while True:
8      if cpx.button_b:
9          cpx.red_led = True
10     else:
11         cpx.red_led = False
12
13  # Can also be written as:
14  #     cpx.red_led = cpx.button_b

```

Listing 11: examples/circuitplayground\_buttons\_1\_neopixel.py

```

1  """This example lights up the third NeoPixel while button A is being pressed, and_
↪lights up the
2  eighth NeoPixel while button B is being pressed."""
3  from adafruit_circuitplayground.express import cpx
4
5  cpx.pixels.brightness = 0.3
6  cpx.pixels.fill((0, 0, 0)) # Turn off the NeoPixels if they're on!
7
8  while True:
9      if cpx.button_a:
10         cpx.pixels[2] = (0, 255, 0)
11     else:

```

(continues on next page)

(continued from previous page)

```

12     cpx.pixels[2] = (0, 0, 0)
13
14     if cpx.button_b:
15         cpx.pixels[7] = (0, 0, 255)
16     else:
17         cpx.pixels[7] = (0, 0, 0)

```

Listing 12: examples/circuitplayground\_buttons\_neopixels.py

```

1  """This example lights up half the NeoPixels red while button A is being pressed, and
   ↪half the
2  NeoPixels green while button B is being pressed."""
3  from adafruit_circuitplayground.express import cpx
4
5  cpx.pixels.brightness = 0.3
6  cpx.pixels.fill((0, 0, 0)) # Turn off the NeoPixels if they're on!
7
8  while True:
9      if cpx.button_a:
10         cpx.pixels[0:5] = [(255, 0, 0)] * 5
11     else:
12         cpx.pixels[0:5] = [(0, 0, 0)] * 5
13
14     if cpx.button_b:
15         cpx.pixels[5:10] = [(0, 255, 0)] * 5
16     else:
17         cpx.pixels[5:10] = [(0, 0, 0)] * 5

```

Listing 13: examples/circuitplayground\_ir\_receive.py

```

1  """THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
2  This example requires the adafruit_irremote.mpy library.
3
4  This example uses the IR receiver found near the center of the board. Works with
   ↪another CPX
5  running the cpx_ir_transmit.py example. The NeoPixels will light up when the buttons
   ↪on the
6  TRANSMITTING CPX are pressed!"""
7  import pulseio
8  import board
9  import adafruit_irremote
10 from adafruit_circuitplayground.express import cpx
11
12 # Create a 'pulseio' input, to listen to infrared signals on the IR receiver
13 pulsein = pulseio.PulseIn(board.IR_RX, maxlen=120, idle_state=True)
14 # Create a decoder that will take pulses and turn them into numbers
15 decoder = adafruit_irremote.GenericDecode()
16
17 while True:
18     cpx.red_led = True
19     pulses = decoder.read_pulses(pulsein)
20     try:
21         # Attempt to convert received pulses into numbers
22         received_code = decoder.decode_bits(pulses, debug=False)
23     except adafruit_irremote.IRNECRepeatException:
24         # We got an unusual short code, probably a 'repeat' signal

```

(continues on next page)

(continued from previous page)

```

25     continue
26 except adafruit_irremote.IRDecodeException:
27     # Something got distorted
28     continue
29
30 print("Infrared code received: ", received_code)
31 if received_code == [66, 84, 78, 65]:
32     print("Button A signal")
33     cpx.pixels.fill((100, 0, 155))
34 if received_code == [66, 84, 78, 64]:
35     print("Button B Signal")
36     cpx.pixels.fill((210, 45, 0))

```

Listing 14: examples/circuitplayground\_ir\_transmit.py

```

1  """THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
2  This example requires the adafruit_irremote.mpy library.
3
4  This example uses the IR transmitter found near the center of the board. Works with
5  ↳ another CPX
6  ↳ running the cpx_ir_receive.py example. Press the buttons to light up the NeoPixels on
7  ↳ the RECEIVING
8  CPX!"""
9  import time
10 import pulseio
11 import board
12 import adafruit_irremote
13 from adafruit_circuitplayground.express import cpx
14
15 # Create a 'pulseio' output, to send infrared signals from the IR transmitter
16 pwm = pulseio.PWMOut(board.IR_TX, frequency=38000, duty_cycle=2 ** 15)
17 pulseout = pulseio.PulseOut(pwm)
18 # Create an encoder that will take numbers and turn them into NEC IR pulses
19 encoder = adafruit_irremote.GenericTransmit(header=[9500, 4500], one=[550, 550],
20 zero=[550, 1700], trail=0)
21
22 while True:
23     if cpx.button_a:
24         print("Button A pressed! \n")
25         cpx.red_led = True
26         encoder.transmit(pulseout, [66, 84, 78, 65])
27         cpx.red_led = False
28         # wait so the receiver can get the full message
29         time.sleep(0.2)
30     if cpx.button_b:
31         print("Button B pressed! \n")
32         cpx.red_led = True
33         encoder.transmit(pulseout, [66, 84, 78, 64])
34         cpx.red_led = False
35         time.sleep(0.2)

```

Listing 15: examples/circuitplayground\_light\_neopixels.py

```

1  """THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
2  This example requires the simpleio.mpy library.
3

```

(continues on next page)



(continued from previous page)

```

4  This example uses the light sensor on the CPX, located next to the picture of the eye
   ↳on the board.
5  Once you have the library loaded, try shining a flashlight on your CPX to watch the
   ↳number of
6  NeoPixels lit up increase, or try covering up the light sensor to watch the number
   ↳decrease."""
7  import time
8  from adafruit_circuitplayground.express import cpx
9  import simpleio
10
11 cpx.pixels.auto_write = False
12 cpx.pixels.brightness = 0.3
13
14 while True:
15     # light value remapped to pixel position
16     peak = simpleio.map_range(cpx.light, 0, 320, 0, 10)
17     print(cpx.light)
18     print(int(peak))
19
20     for i in range(0, 10, 1):
21         if i <= peak:
22             cpx.pixels[i] = (0, 255, 255)
23         else:
24             cpx.pixels[i] = (0, 0, 0)
25     cpx.pixels.show()
26     time.sleep(0.05)

```

Listing 16: examples/circuitplayground\_light.py

```

1  """This example uses the light sensor on your CPX, located next to the picture of the
   ↳eye. Try
2  shining a flashlight on your CPX, or covering the light sensor with your finger to
   ↳see the values
3  increase and decrease."""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  while True:
8      print("Light:", cpx.light)
9      time.sleep(1)

```

Listing 17: examples/circuitplayground\_neopixel\_0\_1.py

```

1  """This example lights up the first and second NeoPixel, red and blue respectively."""
2  from adafruit_circuitplayground.express import cpx
3
4  cpx.pixels.brightness = 0.3
5
6  while True:
7      cpx.pixels[0] = (255, 0, 0)
8      cpx.pixels[1] = (0, 0, 255)

```

Listing 18: examples/circuitplayground\_light\_plotter.py

```

1  """If you're using Mu, this example will plot the light levels from the light sensor,
   ↳ (located next
2  to the eye) on your CPX. Try shining a flashlight on your CPX, or covering the light
   ↳ sensor to see
3  the plot increase and decrease."""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  while True:
8      print("Light:", cpx.light)
9      print((cpx.light,))
10     time.sleep(0.1)

```

Listing 19: examples/circuitplayground\_play\_file\_buttons.py

```

1  """THIS EXAMPLE REQUIRES A WAV FILE FROM THE examples FOLDER IN THE
2  Adafruit_CircuitPython_CircuitPlayground REPO found at:
3  https://github.com/adafruit/Adafruit_CircuitPython_CircuitPlayground/tree/master/
   ↳ examples
4
5  Copy the "dip.wav" and "rise.wav" files to your CIRCUITPY drive.
6
7  Once the files are copied, this example plays a different wav file for each button
   ↳ pressed!"""
8  from adafruit_circuitplayground.express import cpx
9
10 while True:
11     if cpx.button_a:
12         cpx.play_file("dip.wav")
13     if cpx.button_b:
14         cpx.play_file("rise.wav")

```

Listing 20: examples/circuitplayground\_play\_file.py

```

1  """THIS EXAMPLE REQUIRES A WAV FILE FROM THE examples FOLDER IN THE
2  Adafruit_CircuitPython_CircuitPlayground REPO found at:
3  https://github.com/adafruit/Adafruit_CircuitPython_CircuitPlayground/tree/master/
   ↳ examples
4
5  Copy the "dip.wav" file to your CIRCUITPY drive.
6
7  Once the file is copied, this example plays a wav file!"""
8  from adafruit_circuitplayground.express import cpx
9
10 cpx.play_file("dip.wav")

```

Listing 21: examples/circuitplayground\_play\_tone\_buttons.py

```

1  """This example plays a different tone for a duration of 1 second for each button
   ↳ pressed."""
2  from adafruit_circuitplayground.express import cpx
3
4  while True:
5      if cpx.button_a:
6          cpx.play_tone(262, 1)

```

(continues on next page)

(continued from previous page)

```
7     if cpx.button_b:
8         cpx.play_tone(294, 1)
```

Listing 22: examples/circuitplayground\_play\_tone.py

```
1  """This example plays two tones for 1 second each. Note that the tones are not in a
   ↳ loop - this is
2  to prevent them from playing indefinitely!"""
3  from adafruit_circuitplayground.express import cpx
4
5  cpx.play_tone(262, 1)
6  cpx.play_tone(294, 1)
```

Listing 23: examples/circuitplayground\_red\_led\_blinky.py

```
1  """This is the "Hello, world!" of CircuitPython: Blinky! This example blinks the
   ↳ little red LED on
2  and off!"""
3  import time
4  from adafruit_circuitplayground.express import cpx
5
6  while True:
7      cpx.red_led = True
8      time.sleep(0.5)
9      cpx.red_led = False
10     time.sleep(0.5)
```

Listing 24: examples/circuitplayground\_red\_led\_blnky\_short.py

```
1  """This is the "Hello, world!" of CircuitPython: Blinky! This example blinks the
   ↳ little red LED on
2  and off! It's a shorter version of the other Blinky example."""
3  import time
4  from adafruit_circuitplayground.express import cpx
5
6  while True:
7      cpx.red_led = not cpx.red_led
8      time.sleep(0.5)
```

Listing 25: examples/circuitplayground\_red\_led.py

```
1  """This example turns on the little red LED."""
2  from adafruit_circuitplayground.express import cpx
3
4  while True:
5      cpx.red_led = True
```

Listing 26: examples/circuitplayground\_slide\_switch\_red\_led.py

```
1  """This example uses the slide switch to control the little red LED."""
2  from adafruit_circuitplayground.express import cpx
3
4  # This code is written to be readable versus being Pylint compliant.
5  # pylint: disable=simplifiable-if-statement
6
```

(continues on next page)

(continued from previous page)

```

7 while True:
8     if cpx.switch:
9         cpx.red_led = True
10    else:
11        cpx.red_led = False

```

Listing 27: examples/circuitplayground\_slide\_switch\_red\_led\_short.py

```

1 """This example uses the slide switch to control the little red LED. When the switch_
   ↳is to the
2 right it returns False, and when it's to the left, it returns True."""
3 from adafruit_circuitplayground.express import cpx
4
5 while True:
6     cpx.red_led = cpx.switch

```

Listing 28: examples/circuitplayground\_slide\_switch.py

```

1 """This example prints the status of the slide switch. Try moving the switch back and_
   ↳forth to see
2 what's printed to the serial console!"""
3 import time
4 from adafruit_circuitplayground.express import cpx
5
6 while True:
7     print("Slide switch:", cpx.switch)
8     time.sleep(0.1)

```

Listing 29: examples/circuitplayground\_sound\_meter.py

```

1 """This example uses the sound sensor, located next to the picture of the ear on your_
   ↳board, to
2 light up the NeoPixels as a sound meter. Try talking to your CPX or clapping, etc, to_
   ↳see the
3 NeoPixels light up!"""
4 import array
5 import math
6 import audiobusio
7 import board
8 from adafruit_circuitplayground.express import cpx
9
10
11 def constrain(value, floor, ceiling):
12     return max(floor, min(value, ceiling))
13
14
15 def log_scale(input_value, input_min, input_max, output_min, output_max):
16     normalized_input_value = (input_value - input_min) / (input_max - input_min)
17     return output_min + math.pow(normalized_input_value, 0.630957) * (output_max -_
   ↳output_min)
18
19
20 def normalized_rms(values):
21     minbuf = int(sum(values) / len(values))
22     return math.sqrt(sum(float(sample - minbuf) *

```

(continues on next page)

(continued from previous page)

```

23         (sample - minbuf) for sample in values) / len(values))
24
25
26 mic = audiobusio.PDMIn(board.MICROPHONE_CLOCK, board.MICROPHONE_DATA,
27                        sample_rate=16000, bit_depth=16)
28
29 samples = array.array('H', [0] * 160)
30 mic.record(samples, len(samples))
31 input_floor = normalized_rms(samples) + 10
32
33 # Lower number means more sensitive - more LEDs will light up with less sound.
34 sensitivity = 500
35 input_ceiling = input_floor + sensitivity
36
37 peak = 0
38 while True:
39     mic.record(samples, len(samples))
40     magnitude = normalized_rms(samples)
41     print((magnitude,))
42
43     c = log_scale(constrain(magnitude, input_floor, input_ceiling),
44                  input_floor, input_ceiling, 0, 10)
45
46     cpx.pixels.fill((0, 0, 0))
47     for i in range(10):
48         if i < c:
49             cpx.pixels[i] = (i * (255 // 10), 50, 0)
50         if c >= peak:
51             peak = min(c, 10 - 1)
52         elif peak > 0:
53             peak = peak - 1
54         if peak > 0:
55             cpx.pixels[int(peak)] = (80, 0, 255)
56     cpx.pixels.show()

```

Listing 30: examples/circuitplayground\_tap\_red\_led.py

```

1  """This example turns on the little red LED and prints to the serial console when you_
2  ↪double-tap
3  the CPX!"""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  # Change to 1 for detecting a single-tap!
8  cpx.detect_taps = 2
9
10 while True:
11     if cpx.tapped:
12         print("Tapped!")
13         cpx.red_led = True
14         time.sleep(0.1)
15     else:
16         cpx.red_led = False

```

Listing 31: examples/circuitplayground\_temperature\_neopixels.py

```

1  """THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
2  This example requires the simpleio.mpy library.
3
4  This example use the temperature sensor on the CPX, located next to the picture of
5  ↳the thermometer
6  on the board. Try warming up the board to watch the number of NeoPixels lit up
7  ↳increase, or cooling
8  it down to see the number decrease. You can set the min and max temperatures to make
9  ↳it more or
10 less sensitive to temperature changes.
11 """
12
13 import time
14 from adafruit_circuitplayground.express import cpx
15 import simpleio
16
17 cpx.pixels.auto_write = False
18 cpx.pixels.brightness = 0.3
19
20 # Set these based on your ambient temperature in Celsius for best results!
21 minimum_temp = 24
22 maximum_temp = 30
23
24 while True:
25     # temperature value remapped to pixel position
26     peak = simpleio.map_range(cpx.temperature, minimum_temp, maximum_temp, 0, 10)
27     print(cpx.temperature)
28     print(int(peak))
29
30     for i in range(0, 10, 1):
31         if i <= peak:
32             cpx.pixels[i] = (0, 255, 255)
33         else:
34             cpx.pixels[i] = (0, 0, 0)
35     cpx.pixels.show()
36     time.sleep(0.05)

```

Listing 32: examples/circuitplayground\_temperature\_plotter.py

```

1  """If you're using Mu, this example will plot the temperature in C and F on the
2  ↳plotter! Click
3  "Plotter" to open it, and place your finger over the sensor to see the numbers change.
4  ↳ The
5  sensor is located next to the picture of the thermometer on the CPX."""
6
7  import time
8  from adafruit_circuitplayground.express import cpx
9
10 while True:
11     print("Temperature C:", cpx.temperature)
12     print("Temperature F:", cpx.temperature * 1.8 + 32)
13     print((cpx.temperature, cpx.temperature * 1.8 + 32))
14     time.sleep(0.1)

```

Listing 33: examples/circuitplayground\_temperature.py

```

1  """This example uses the temperature sensor on the CPX, located next to the image of
   ↳ a thermometer
2  on the board. It prints the temperature in both C and F to the serial console. Try
   ↳ putting your
3  finger over the sensor to see the numbers change!"""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  while True:
8      print("Temperature C:", cpx.temperature)
9      print("Temperature F:", cpx.temperature * 1.8 + 32)
10     time.sleep(1)

```

Listing 34: examples/circuitplayground\_touch\_pixel\_fill\_rainbow.py

```

1  """This example uses the capacitive touch pads on the CPX. They are located around
   ↳ the outer edge
2  of the board and are labeled A1-A7. (A0 is not a touch pad.) This example lights up
   ↳ all the
3  NeoPixels a different color of the rainbow for each pad touched!"""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  cpx.pixels.brightness = 0.3
8
9  while True:
10     if cpx.touch_A1:
11         print("Touched A1!")
12         cpx.pixels.fill((255, 0, 0))
13     if cpx.touch_A2:
14         print("Touched A2!")
15         cpx.pixels.fill((210, 45, 0))
16     if cpx.touch_A3:
17         print("Touched A3!")
18         cpx.pixels.fill((155, 100, 0))
19     if cpx.touch_A4:
20         print("Touched A4!")
21         cpx.pixels.fill((0, 255, 0))
22     if cpx.touch_A5:
23         print("Touched A5!")
24         cpx.pixels.fill((0, 135, 125))
25     if cpx.touch_A6:
26         print("Touched A6!")
27         cpx.pixels.fill((0, 0, 255))
28     if cpx.touch_A7:
29         print("Touched A7!")
30         cpx.pixels.fill((100, 0, 155))
31     time.sleep(0.1)

```

Listing 35: examples/circuitplayground\_touch\_pixel\_rainbow.py

```

1  """This example uses the capacitive touch pads on the CPX. They are located around
   ↳ the outer edge
2  of the board and are labeled A1-A7. (A0 is not a touch pad.) This example lights up
   ↳ the nearest

```

(continues on next page)

(continued from previous page)

```

3  NeoPixel to that pad a different color of the rainbow!"""
4  import time
5  from adafruit_circuitplayground.express import cpx
6
7  cpx.pixels.brightness = 0.3
8
9  while True:
10     if cpx.touch_A1:
11         print("Touched A1!")
12         cpx.pixels[6] = (255, 0, 0)
13     if cpx.touch_A2:
14         print("Touched A2!")
15         cpx.pixels[8] = (210, 45, 0)
16     if cpx.touch_A3:
17         print("Touched A3!")
18         cpx.pixels[9] = (155, 100, 0)
19     if cpx.touch_A4:
20         print("Touched A4!")
21         cpx.pixels[0] = (0, 255, 0)
22     if cpx.touch_A5:
23         print("Touched A5!")
24         cpx.pixels[1] = (0, 135, 125)
25     if cpx.touch_A6:
26         print("Touched A6!")
27         cpx.pixels[3] = (0, 0, 255)
28     if cpx.touch_A7:
29         print("Touched A7!")
30         cpx.pixels[4] = (100, 0, 155)
31     time.sleep(0.1)

```

## 5.2 adafruit\_circuitplayground.express

CircuitPython driver from [CircuitPlayground Express](#).

- Author(s): Kattni Rembor, Scott Shawcroft

**class** `adafruit_circuitplayground.express.Express`

Represents a single CircuitPlayground Express. Do not use more than one at a time.

**acceleration**

Obtain data from the x, y and z axes.





This example prints the values. Try moving the board to see how the printed values change.

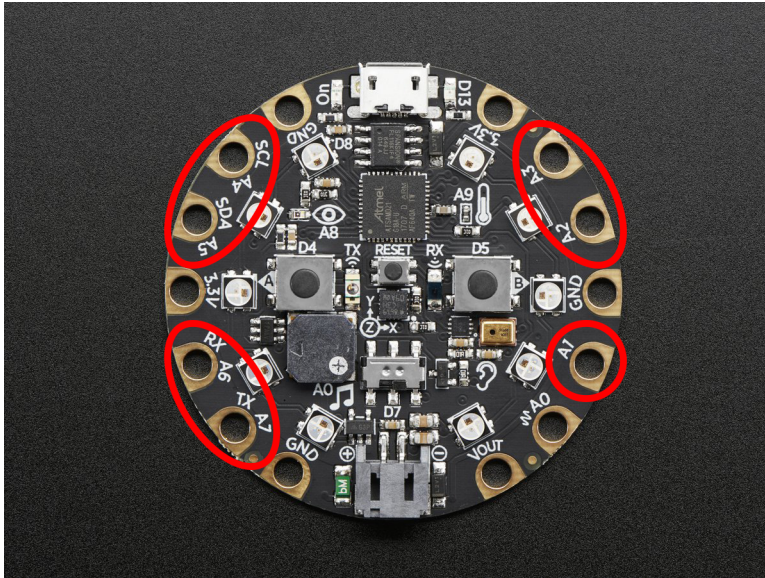
```
from adafruit_circuitplayground.express import cpx

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

#### **adjust\_touch\_threshold** (*adjustment*)

Adjust the threshold needed to activate the capacitive touch pads. Higher numbers make the touch pads less sensitive.

**Parameters** **adjustment** (*int*) – The desired threshold increase



```
from adafruit_circuitplayground.express import cpx

cpx.adjust_touch_threshold(200)
```

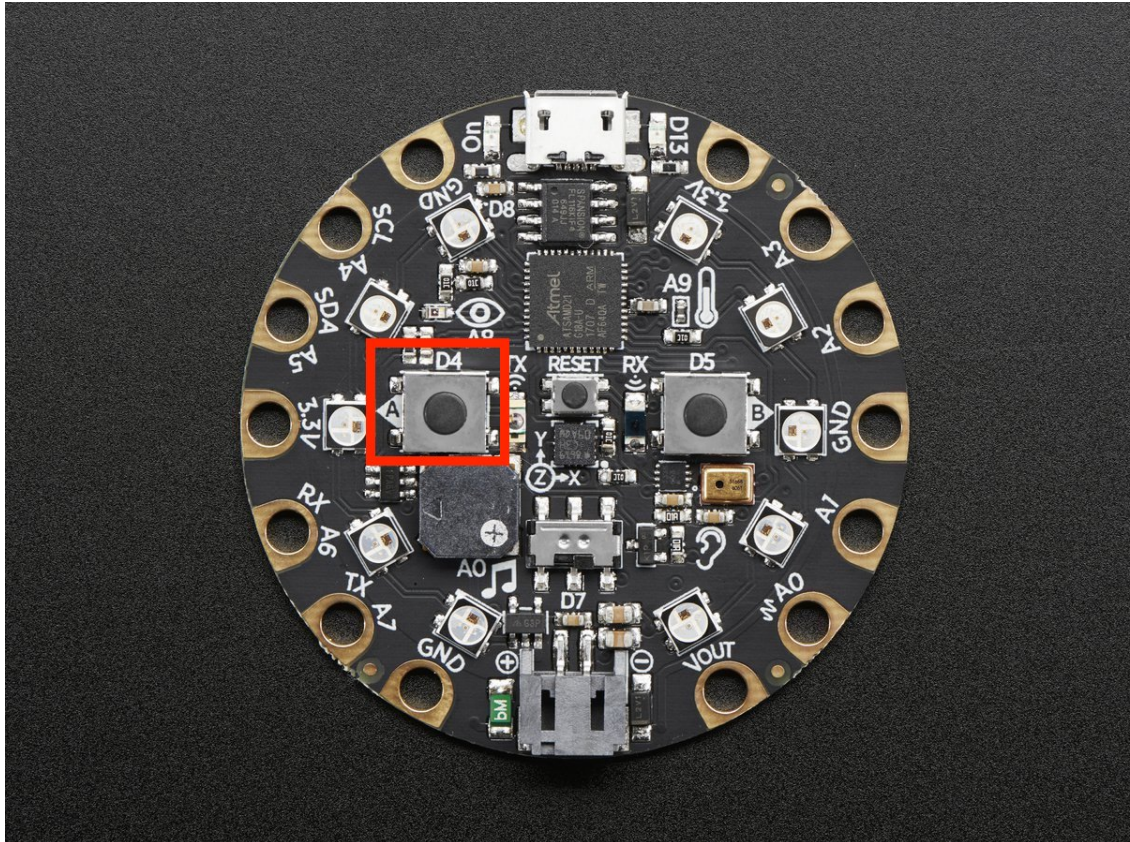
(continues on next page)

(continued from previous page)

```
while True:
    if cpx.touch_A1:
        print('Touched pad A1')
```

**button\_a**

True when Button A is pressed. False if not.



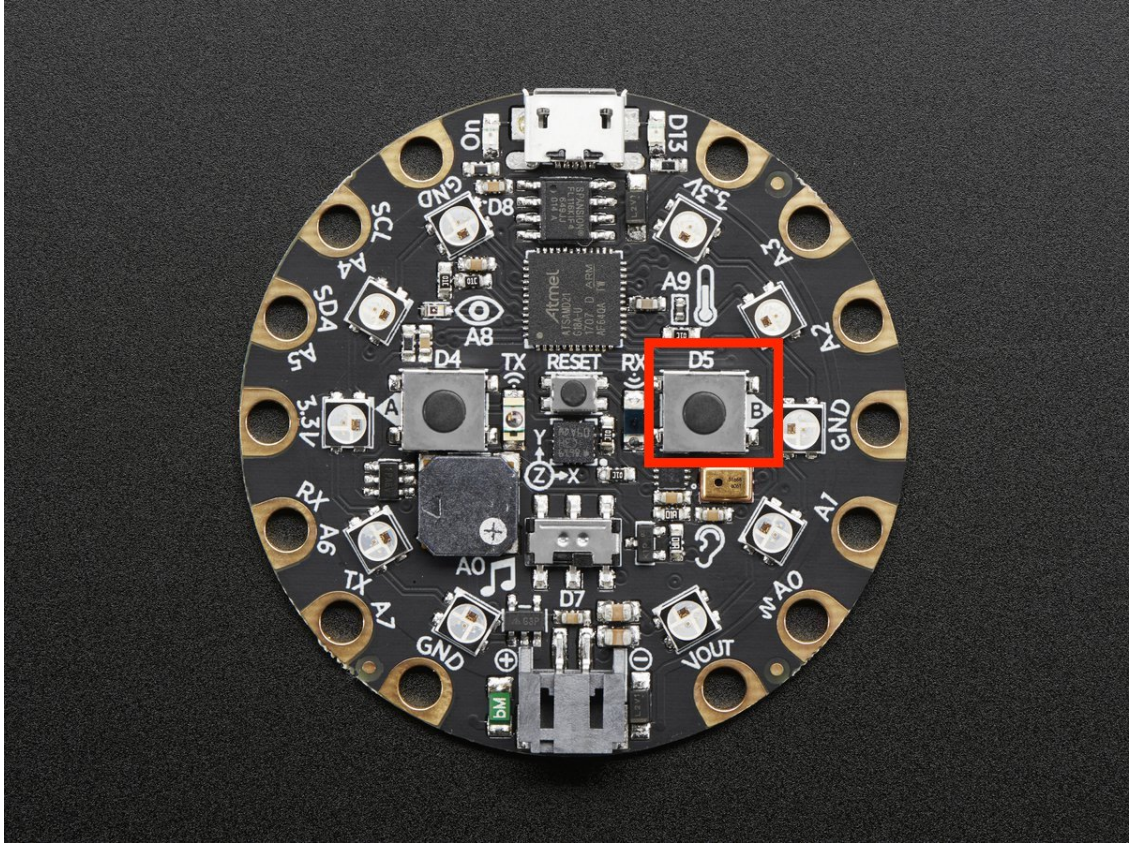
```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.button_a:
        print("Button A pressed!")
```

**button\_b**

True when Button B is pressed. False if not.



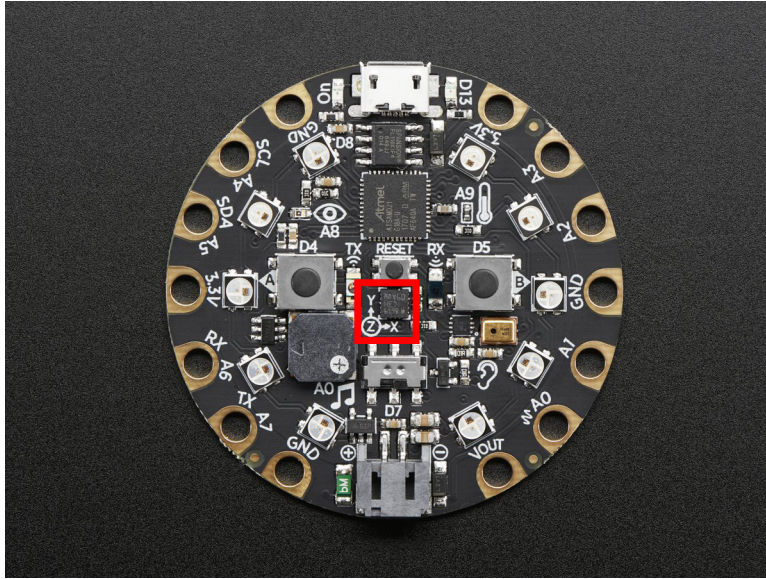


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.button_b:
        print("Button B pressed!")
```

#### **detect\_taps**

Configure what type of tap is detected by `cpx.tapped`. Use 1 for single-tap detection and 2 for double-tap detection. This does nothing without `cpx.tapped`.



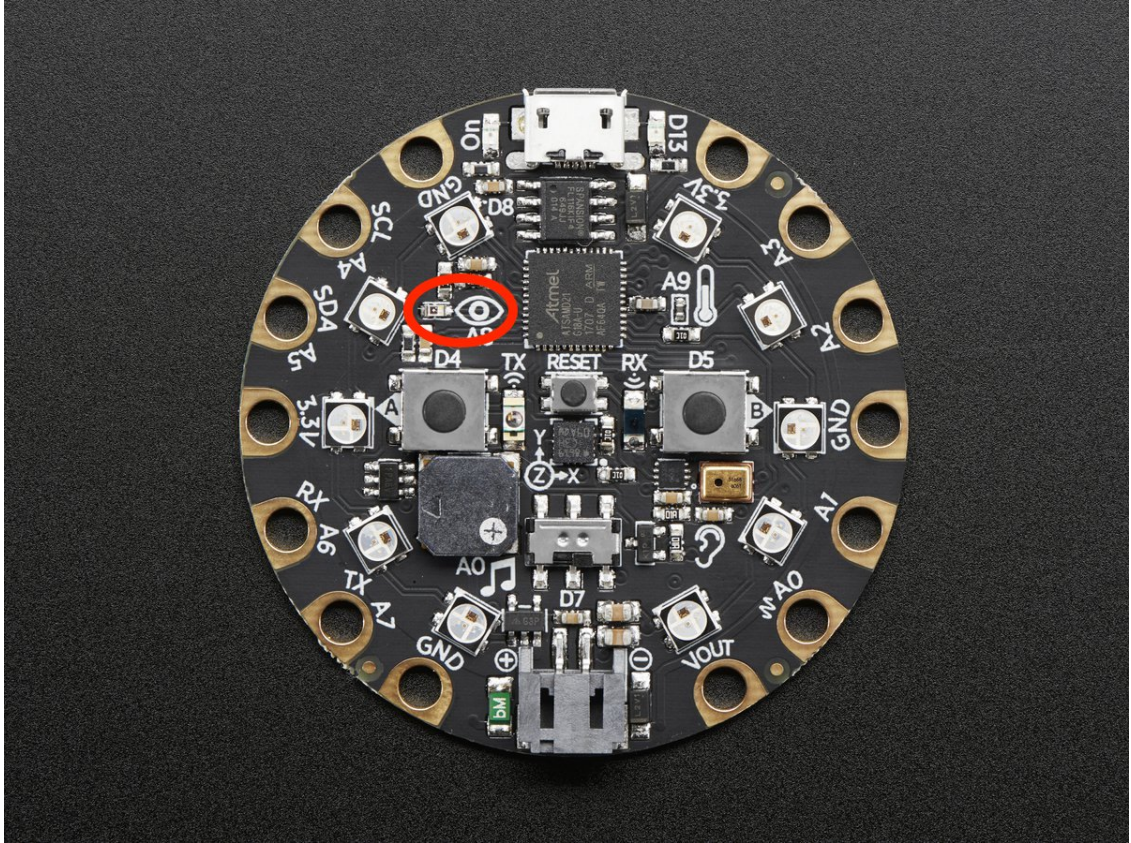
```
from adafruit_circuitplayground.express import cpx

cpx.detect_taps = 1
while True:
    if cpx.tapped:
        print("Single tap detected!")
```

### light

The brightness of the CircuitPlayground in approximate Lux.





Try covering the sensor next to the eye to see it change.

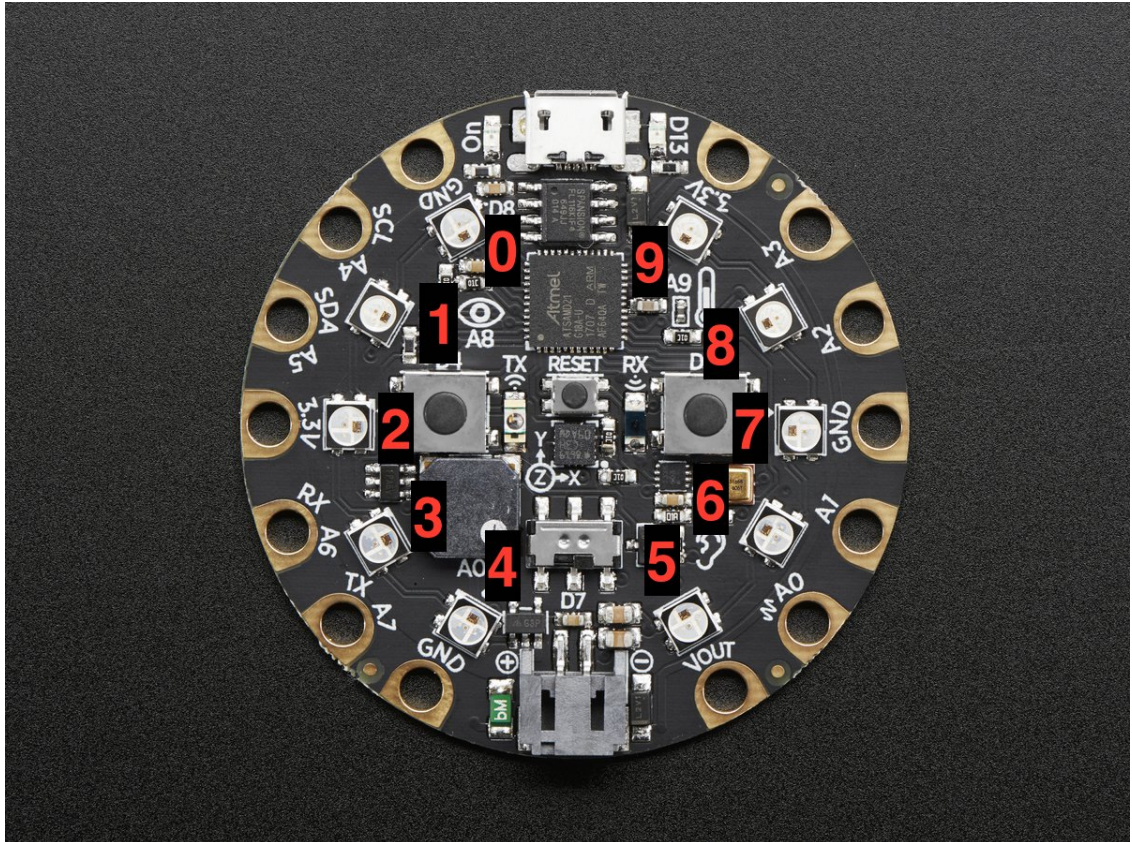
```
from adafruit_circuitplayground.express import cpx
import time

while True:
    print("Lux:", cpx.light)
    time.sleep(1)
```

### pixels

Sequence-like object representing the ten NeoPixels around the outside of the CircuitPlayground. Each pixel is at a certain index in the sequence as labeled below. Colors can be RGB hex like 0x110000 for red where each two digits are a color (0xRRGGBB) or a tuple like (17, 0, 0) where (R, G, B). Set the global brightness using any number from 0 to 1 to represent a percentage, i.e. 0.3 sets global brightness to 30%.

See `neopixel.NeoPixel` for more info.



Here is an example that sets the first pixel green and the second red.

```
from adafruit_circuitplayground.express import cpx

cpx.pixels.brightness = 0.3
cpx.pixels[0] = 0x003000
cpx.pixels[9] = (30, 0, 0)

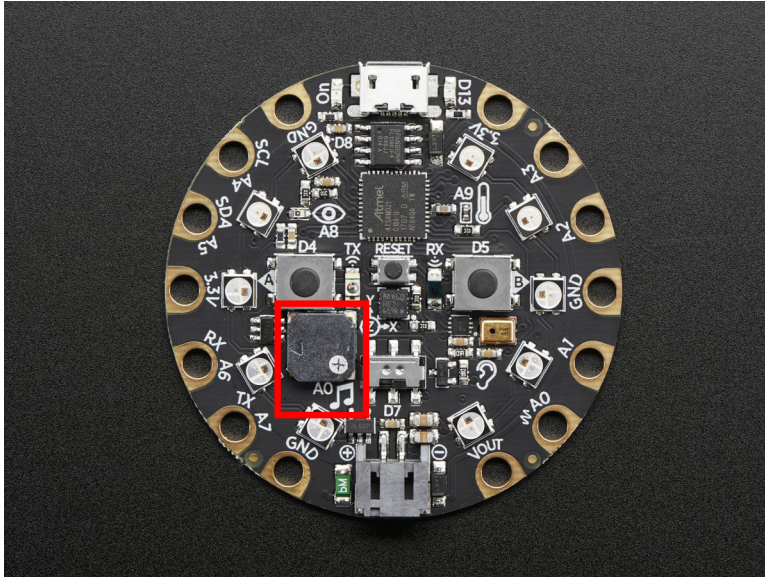
# Wait forever. CTRL-C to quit.
while True:
    pass
```

**play\_file**(*file\_name*)

Play a .wav file using the onboard speaker.

**Parameters** *file\_name* – The name of your .wav file in quotation marks including .wav





```
from adafruit_circuitplayground.express import cpx

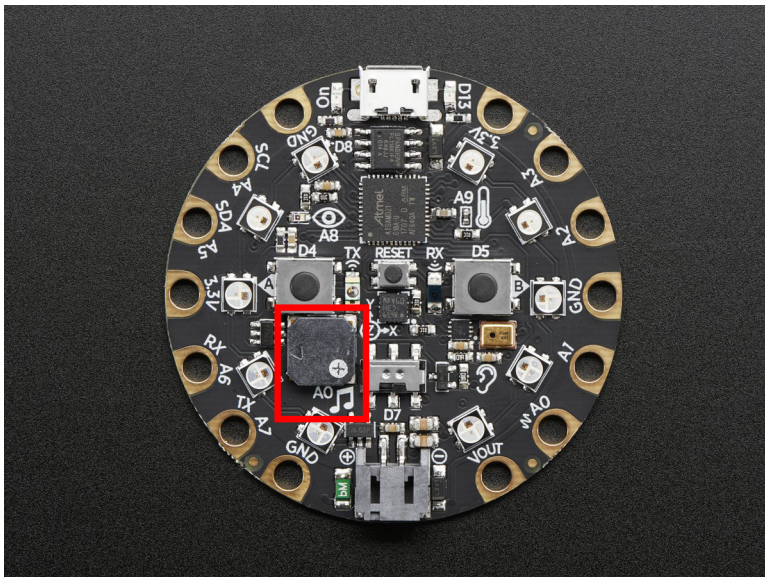
while True:
    if cpx.button_a:
        cpx.play_file("laugh.wav")
    elif cpx.button_b:
        cpx.play_file("rimshot.wav")
```

**play\_tone** (*frequency*, *duration*)

Produce a tone using the speaker. Try changing frequency to change the pitch of the tone.

#### Parameters

- **frequency** (*int*) – The frequency of the tone in Hz
- **duration** (*float*) – The duration of the tone in seconds

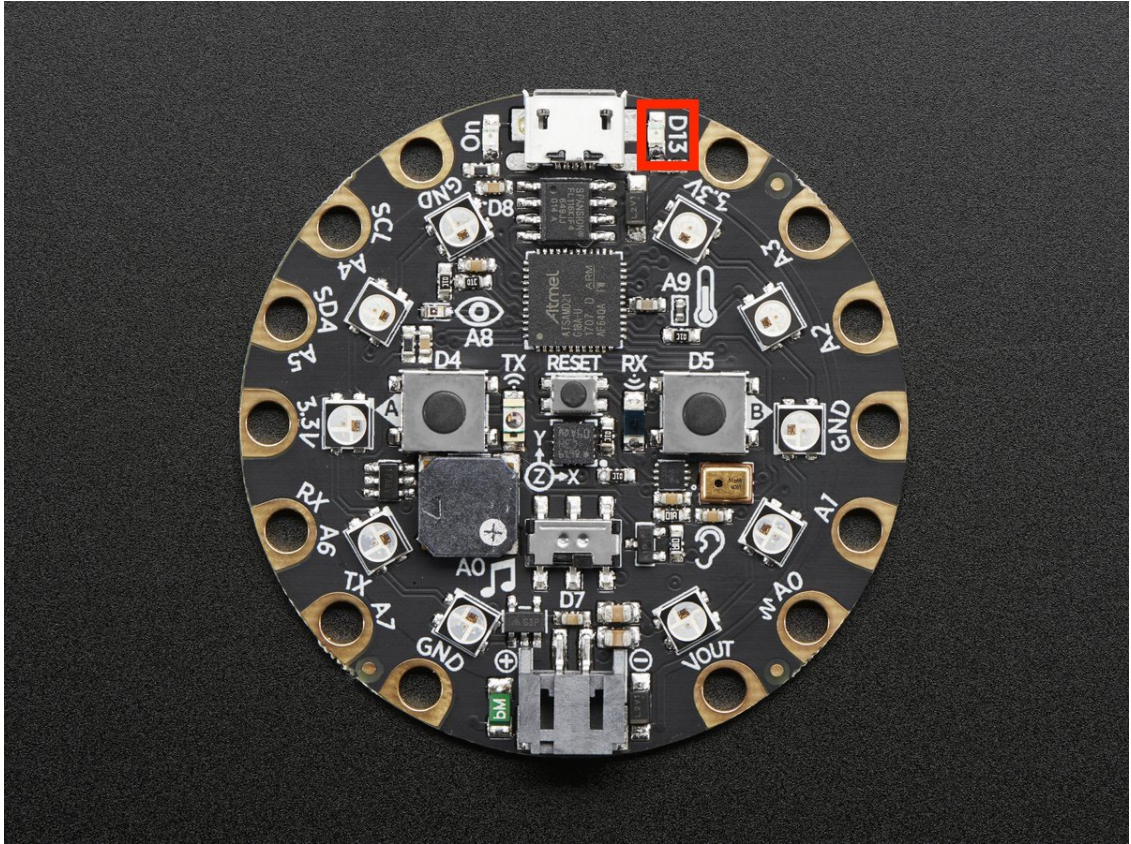


```
from adafruit_circuitplayground.express import cpx

cpx.play_tone(440, 1)
```

**red\_led**

The red led next to the USB plug marked D13.



```
from adafruit_circuitplayground.express import cpx
import time

while True:
    cpx.red_led = True
    time.sleep(1)
    cpx.red_led = False
    time.sleep(1)
```

**shake** (*shake\_threshold=30*)

Detect when device is shaken.

**Parameters** **shake\_threshold** (*int*) – The threshold shake must exceed to return true (Default: 30)





```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.shake():
        print("Shake detected!")
```

Decreasing `shake_threshold` increases shake sensitivity, i.e. the code will return a shake detected more easily with a lower `shake_threshold`. Increasing it causes the opposite. `shake_threshold` requires a minimum value of 10 - 10 is the value when the board is not moving, therefore anything less than 10 will erroneously report a constant shake detected.

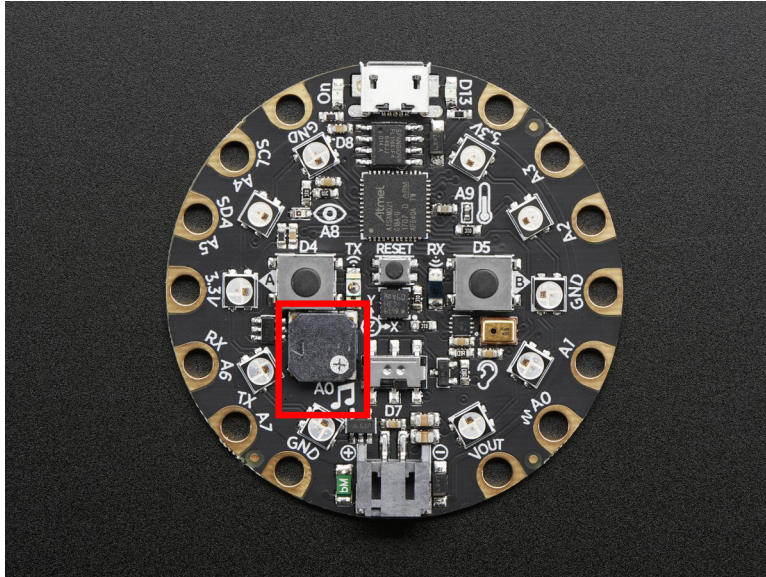
```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.shake(shake_threshold=20):
        print("Shake detected more easily than before!")
```

#### **start\_tone** (*frequency*)

Produce a tone using the speaker. Try changing frequency to change the pitch of the tone.

**Parameters** **frequency** (*int*) – The frequency of the tone in Hz

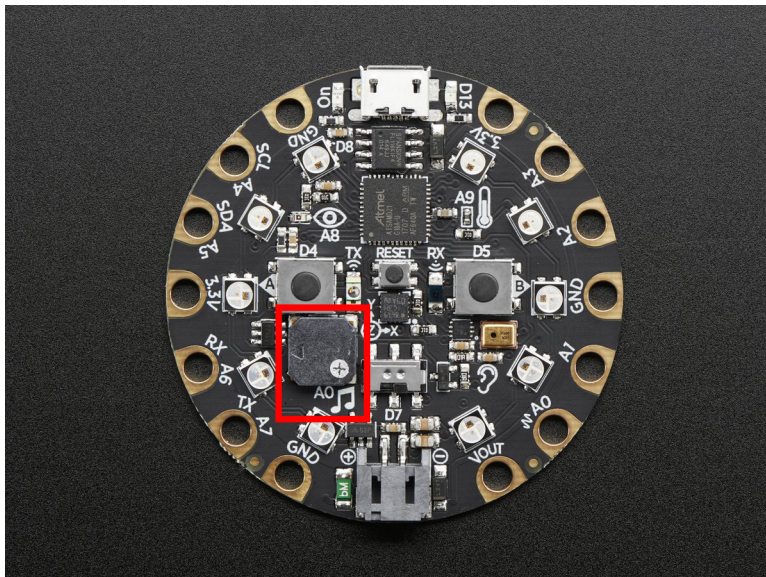


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.button_a:
        cpx.start_tone(262)
    elif cpx.button_b:
        cpx.start_tone(294)
    else:
        cpx.stop_tone()
```

**stop\_tone()**

Use with start\_tone to stop the tone produced.



```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.button_a:
```

(continues on next page)



(continued from previous page)

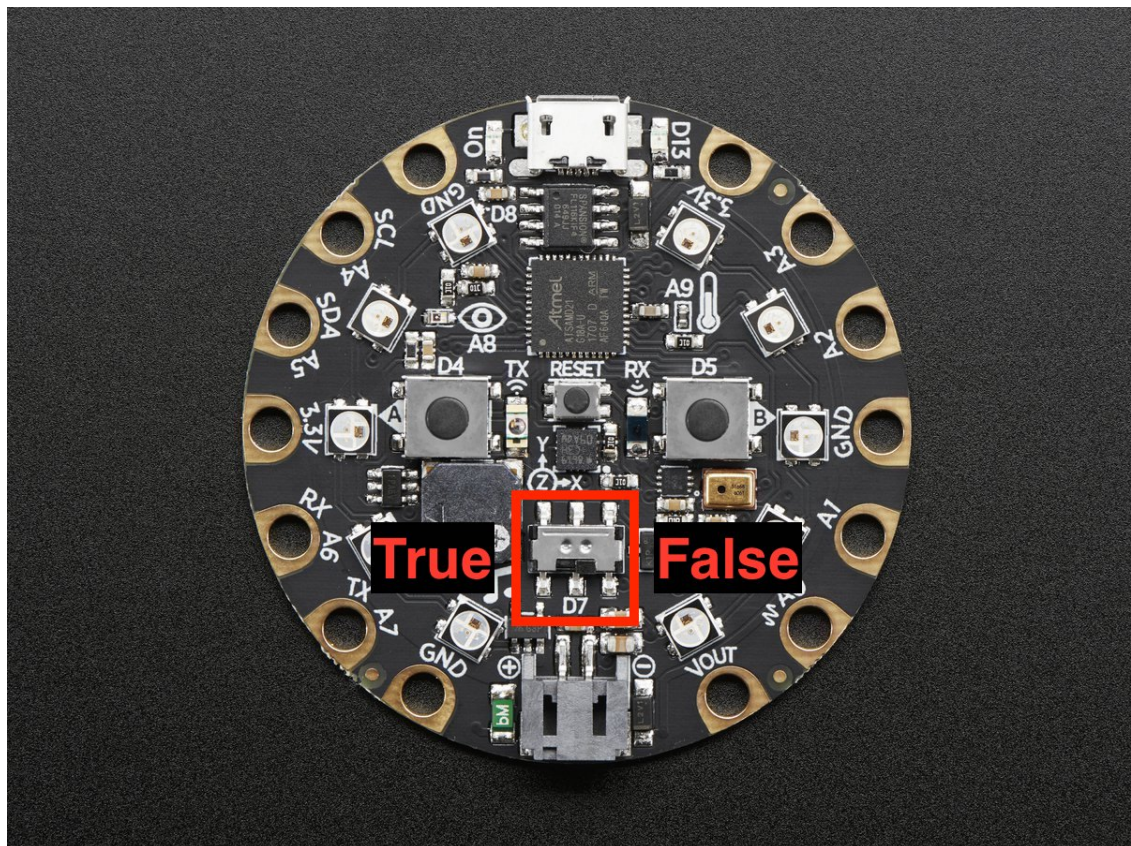
```

        cpx.start_tone(262)
    elif cpx.button_b:
        cpx.start_tone(294)
    else:
        cpx.stop_tone()

```

switch

True when the switch is to the left next to the music notes. False when it is to the right towards the ear.

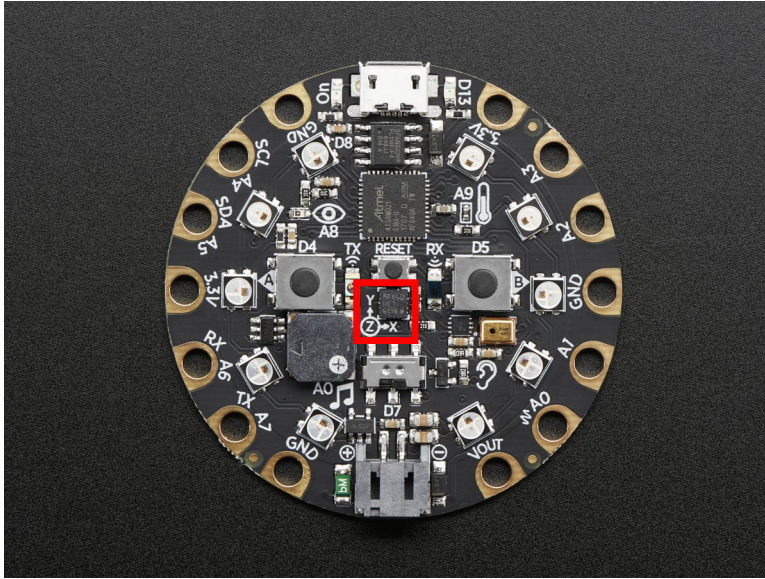


```
from adafruit_circuitplayground.express import cpx
import time

while True:
    print("Slide switch:", cpx.switch)
    time.sleep(1)
```

tapped

True once after detecting a tap. Requires `cpx.detect_taps`.



Tap the CPX once for a single-tap, or quickly tap twice for a double-tap.

```
from adafruit_circuitplayground.express import cpx

cpx.detect_taps = 1

while True:
    if cpx.tapped:
        print("Single tap detected!")
```

To use single and double tap together, you must have a delay between them. It will not function properly without it. This example uses both by counting a specified number of each type of tap before moving on in the code.

```
from adafruit_circuitplayground.express import cpx

# Set to check for single-taps.
cpx.detect_taps = 1
tap_count = 0

# We're looking for 2 single-taps before moving on.
while tap_count < 2:
    if cpx.tapped:
        tap_count += 1
print("Reached 2 single-taps!")

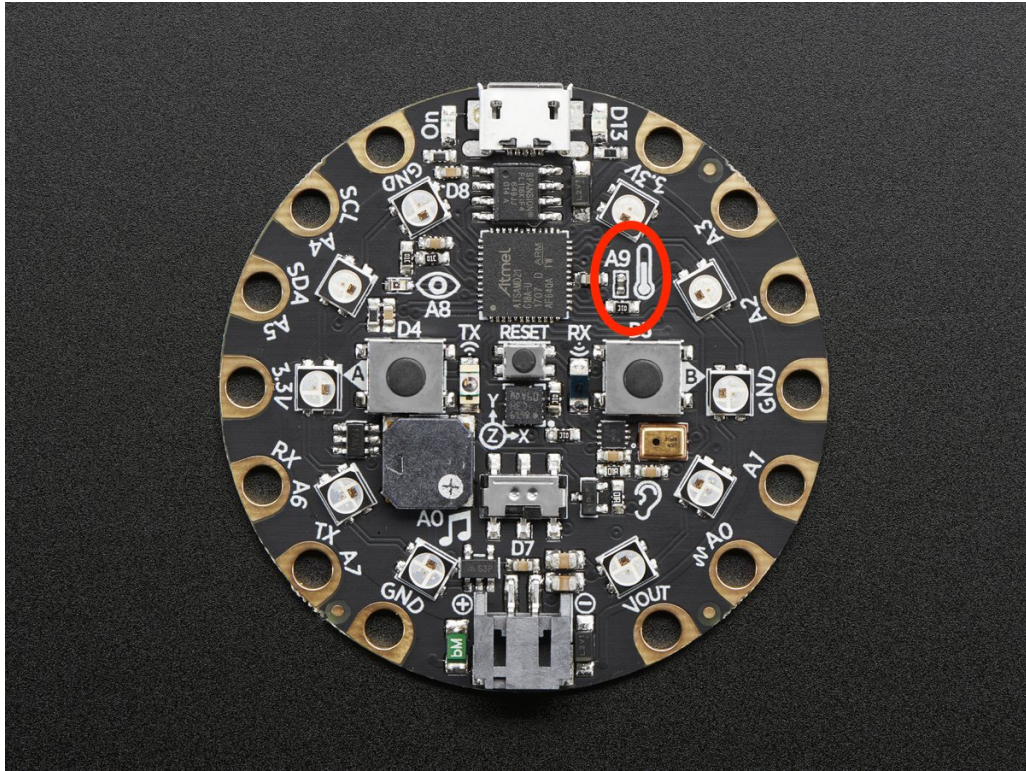
# Now switch to checking for double-taps
tap_count = 0
cpx.detect_taps = 2

# We're looking for 2 double-taps before moving on.
while tap_count < 2:
    if cpx.tapped:
        tap_count += 1
print("Reached 2 double-taps!")
print("Done.")
```

**temperature**



The temperature of the CircuitPlayground in Celsius.



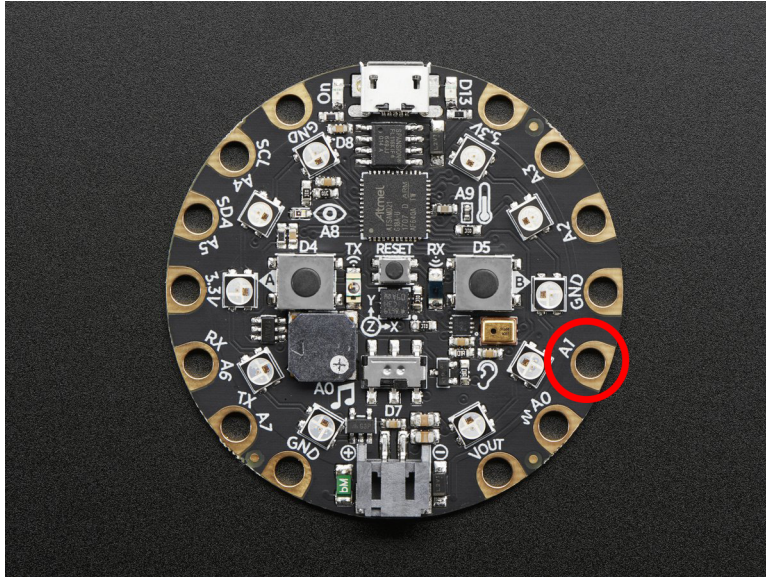
Converting this to Farenheit is easy!

```
from adafruit_circuitplayground.express import cpx
import time

while True:
    temperature_c = cpx.temperature
    temperature_f = temperature_c * 1.8 + 32
    print("Temperature celsius:", temperature_c)
    print("Temperature fahrenheit:", temperature_f)
    time.sleep(1)
```

#### **touch\_A1**

Detect touch on capacitive touch pad A1.

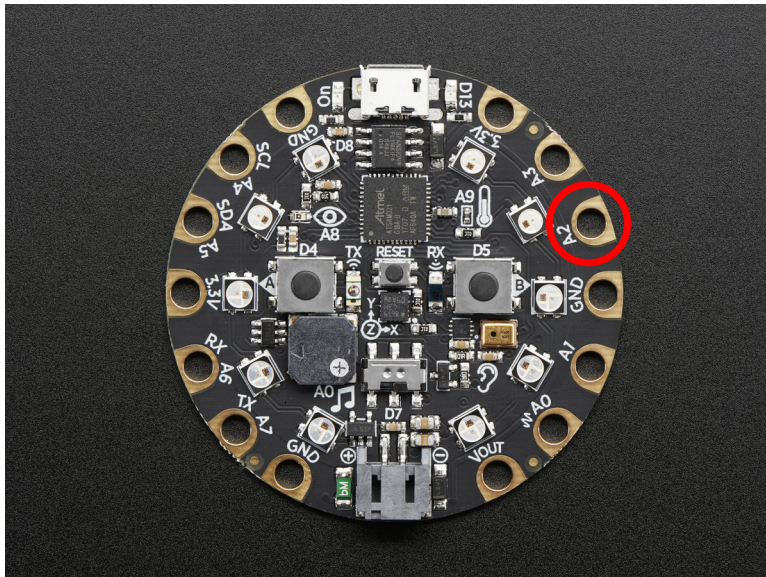


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A1:
        print('Touched pad A1')
```

#### touch\_A2

Detect touch on capacitive touch pad A2.



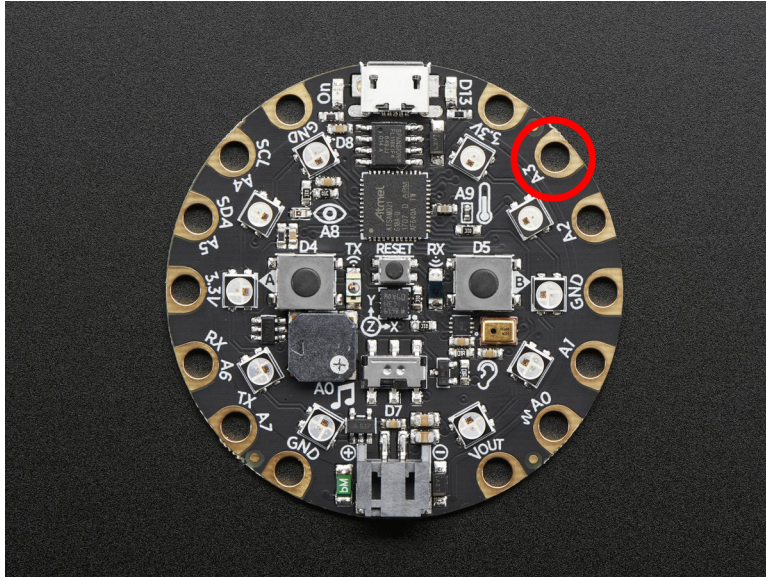
```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A2:
        print('Touched pad A2')
```

#### touch\_A3

Detect touch on capacitive touch pad A3.



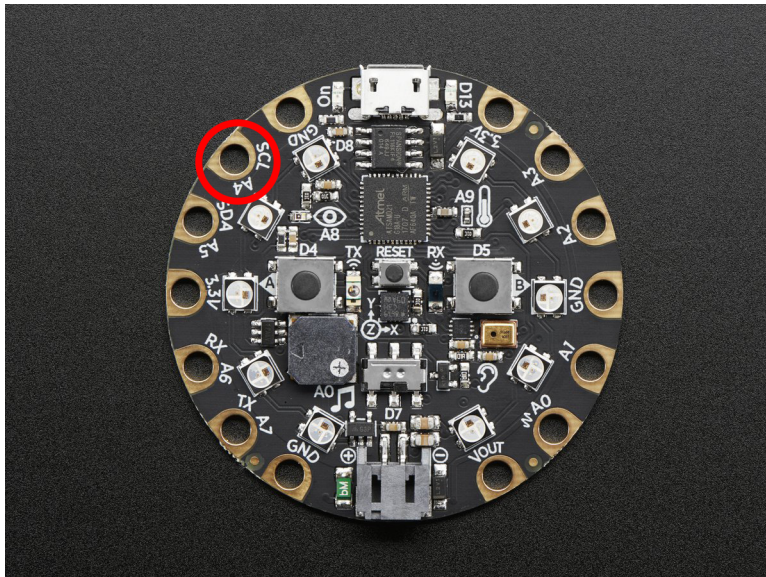


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A3:
        print('Touched pad A3')
```

#### touch\_A4

Detect touch on capacitive touch pad A4.

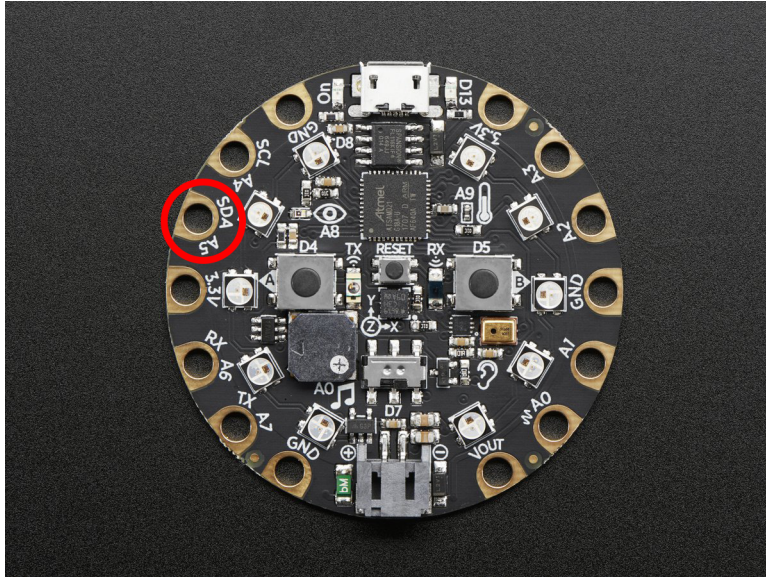


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A4:
        print('Touched pad A4')
```

#### touch\_A5

Detect touch on capacitive touch pad A5.

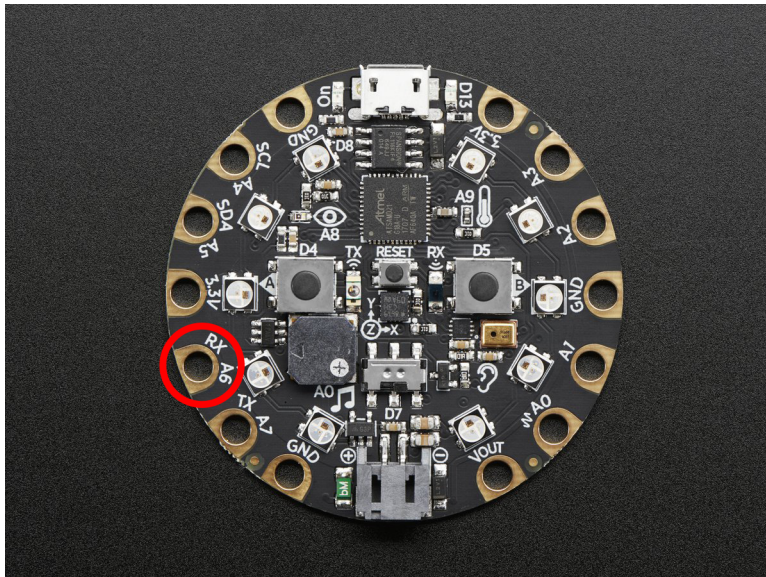


```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A5:
        print('Touched pad A5')
```

#### touch\_A6

Detect touch on capacitive touch pad A6.



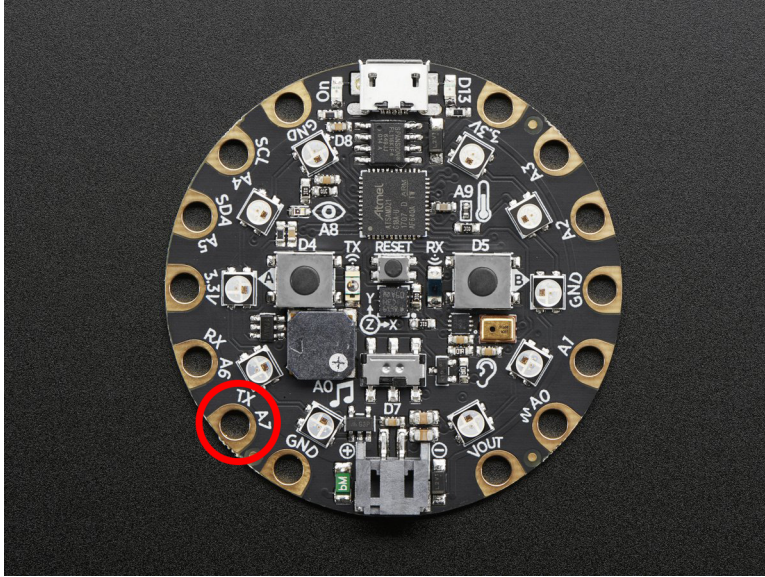
```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A6:
        print('Touched pad A6')
```

#### touch\_A7

Detect touch on capacitive touch pad A7.





```
from adafruit_circuitplayground.express import cpx

while True:
    if cpx.touch_A7:
        print('Touched pad A7')
```

**class** `adafruit_circuitplayground.express.Photocell` (*pin*)

Simple driver for analog photocell on the CircuitPlayground Express.

**light**

Light level in SI Lux.

`adafruit_circuitplayground.express.cpx` = `<adafruit_circuitplayground.express.Express object>`

Object that is automatically created on import.

To use, simply import it from the module:

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



## CHAPTER 6

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



### **a**

`adafruit_circuitplayground.express`, [24](#)



## A

`acceleration` (*adafruit\_circuitplayground.express.Express attribute*), 24  
`adafruit_circuitplayground.express` (module), 24  
`adjust_touch_threshold()` (*adafruit\_circuitplayground.express.Express method*), 25

## B

`button_a` (*adafruit\_circuitplayground.express.Express attribute*), 26  
`button_b` (*adafruit\_circuitplayground.express.Express attribute*), 26

## C

`cpx` (in module *adafruit\_circuitplayground.express*), 41

## D

`detect_taps` (*adafruit\_circuitplayground.express.Express attribute*), 27

## E

`Express` (class in *adafruit\_circuitplayground.express*), 24

## L

`light` (*adafruit\_circuitplayground.express.Express attribute*), 28  
`light` (*adafruit\_circuitplayground.express.Photocell attribute*), 41

## P

`Photocell` (class in *adafruit\_circuitplayground.express*), 41  
`pixels` (*adafruit\_circuitplayground.express.Express attribute*), 29  
`play_file()` (*adafruit\_circuitplayground.express.Express method*), 30

`play_tone()` (*adafruit\_circuitplayground.express.Express method*), 31

## R

`red_led` (*adafruit\_circuitplayground.express.Express attribute*), 32

## S

`shake()` (*adafruit\_circuitplayground.express.Express method*), 32  
`start_tone()` (*adafruit\_circuitplayground.express.Express method*), 33  
`stop_tone()` (*adafruit\_circuitplayground.express.Express method*), 34  
`switch` (*adafruit\_circuitplayground.express.Express attribute*), 35

## T

`tapped` (*adafruit\_circuitplayground.express.Express attribute*), 35  
`temperature` (*adafruit\_circuitplayground.express.Express attribute*), 36  
`touch_A1` (*adafruit\_circuitplayground.express.Express attribute*), 37  
`touch_A2` (*adafruit\_circuitplayground.express.Express attribute*), 38  
`touch_A3` (*adafruit\_circuitplayground.express.Express attribute*), 38  
`touch_A4` (*adafruit\_circuitplayground.express.Express attribute*), 39  
`touch_A5` (*adafruit\_circuitplayground.express.Express attribute*), 39  
`touch_A6` (*adafruit\_circuitplayground.express.Express attribute*), 40  
`touch_A7` (*adafruit\_circuitplayground.express.Express attribute*), 40