
Adafruit Circuit Playground Library Documentation

Release 1.0

Scott Shawcroft

Feb 24, 2018

Contents

1	Installation	3
2	Usage Example	5
3	Contributing	7
4	Building locally	9
4.1	Sphinx documentation	9
5	Table of Contents	11
5.1	Simple test	11
5.2	adafruit_circuitplayground.express	13
6	Indices and tables	31
	Python Module Index	33

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 5.1: examples/acceleration_simpletest.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 5.2: examples/pixels_simpletest.py

```
1 # CircuitPython demo - NeoPixel
2
3 import time
4 from adafruit_circuitplayground.express import cpx
5
6
7 # The number of pixels in the strip
8 numpix = 10
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)
17     elif pos < 170:
```

```

18         pos -= 85
19         return (int(255 - pos*3), 0, int(pos*3))
20     #else:
21     pos -= 170
22     return (0, int(pos*3), int(255 - pos*3))
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 cpx.pixels.auto_write = False
33 cpx.pixels.brightness = 0.3
34 while True:
35     rainbow_cycle(0.001)    # rainbowcycle with 1ms delay per step

```

Listing 5.3: examples/shake_simpletest.py

```

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

```

Listing 5.4: examples/tapdetect_simpletest.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 5.5: examples/tapdetectsimple_simpletest.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 5.6: examples/tone_simpletest.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 5.7: examples/touched_simpletest.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')

```

5.2 adafruit_circuitplayground.express

CircuitPython driver from [CircuitPlayground Express](#).

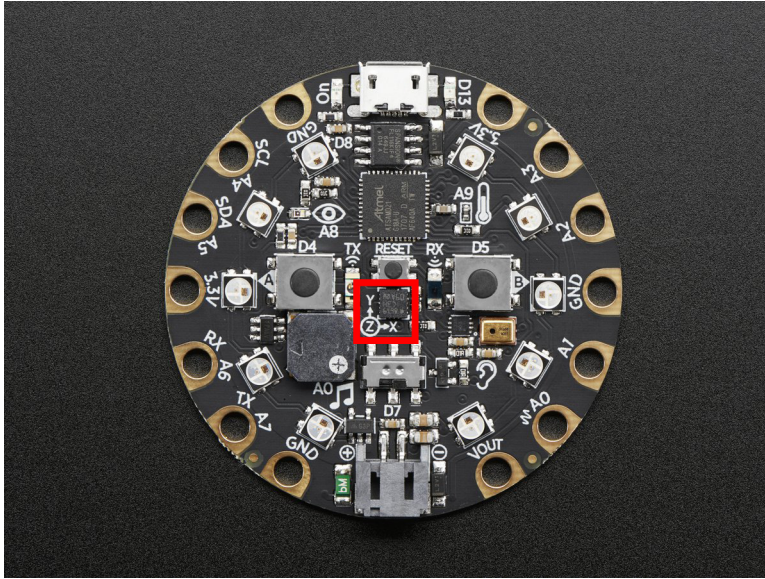
- Author(s): Kattni Rembor, Scott Shawcroft

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.

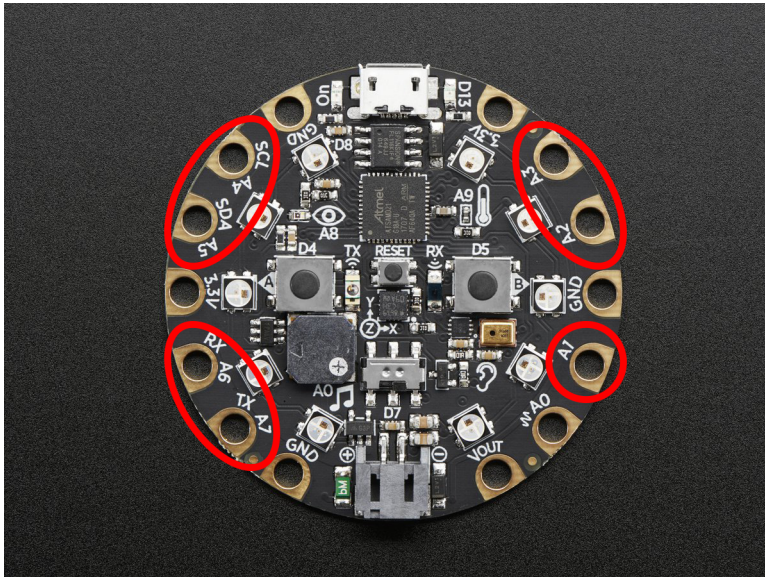
```
from adafruit_circuitplayground.express import cpx

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



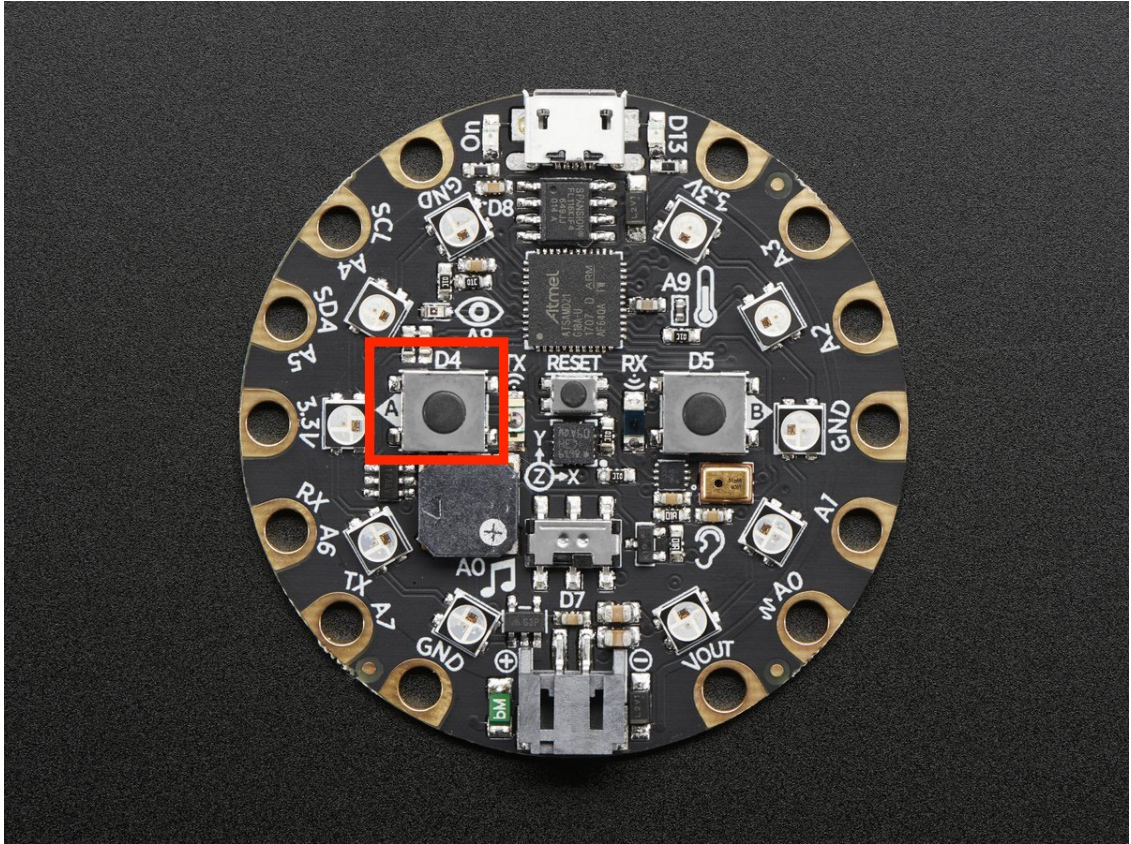
```
from adafruit_circuitplayground.express import cpx

cpx.adjust_touch_threshold(200)
```

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

button_a

True when Button A is pressed. False if not.

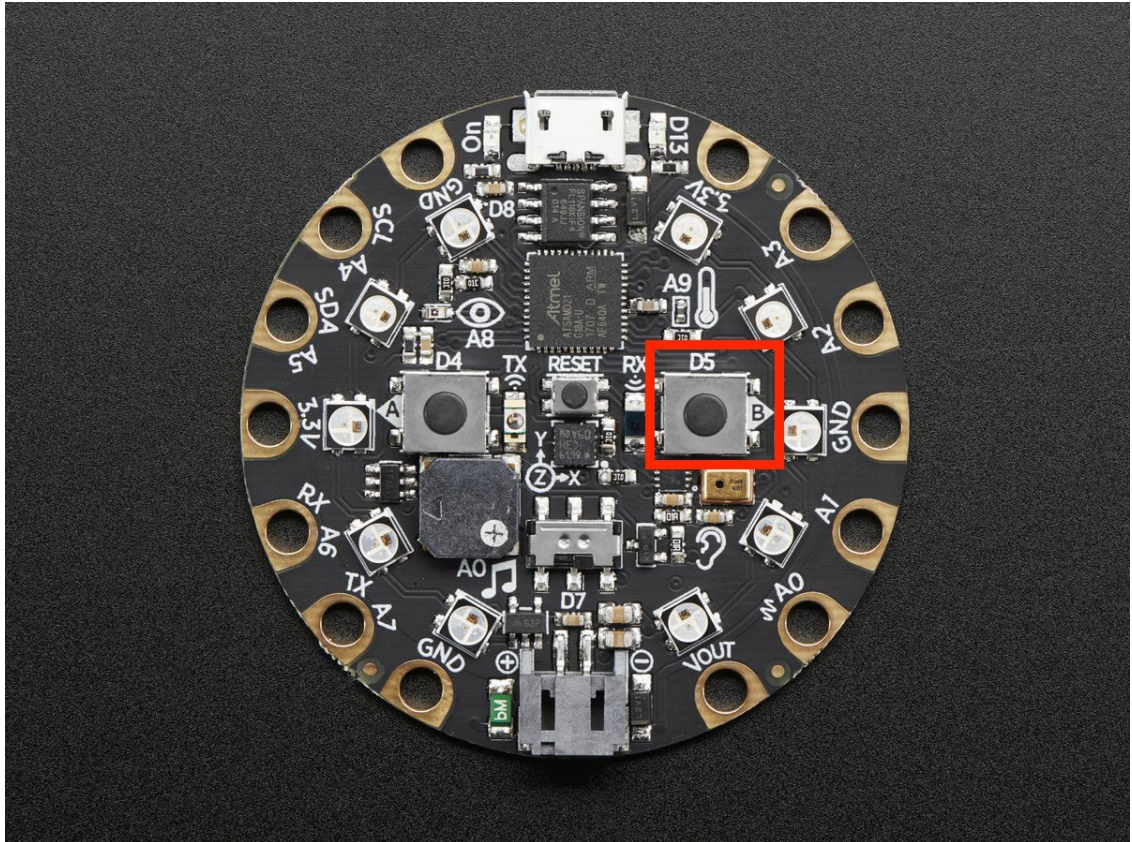


```
from adafruit_circuitplayground.express import cpx

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

button_b

True when Button B is pressed. False if not.

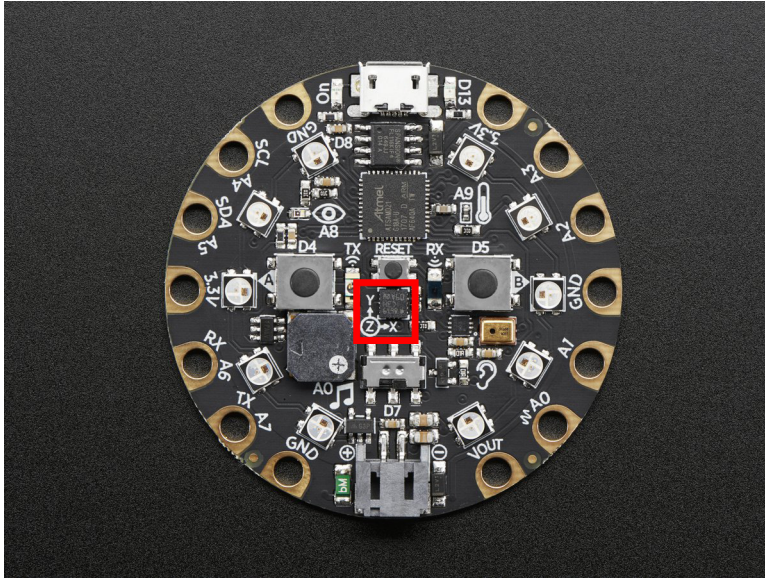


```
from adafruit_circuitplayground.express import cpx

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`.

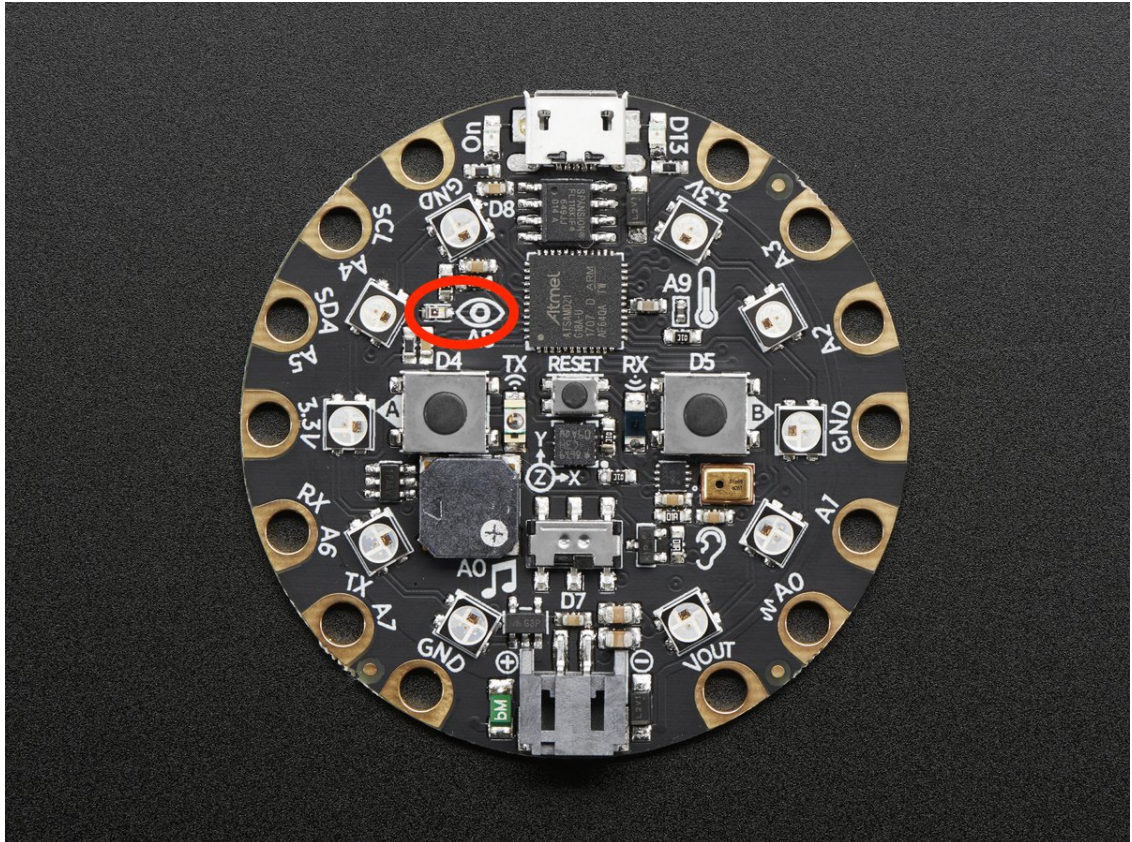


```
from adafruit_circuitplayground.express import cpx

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.

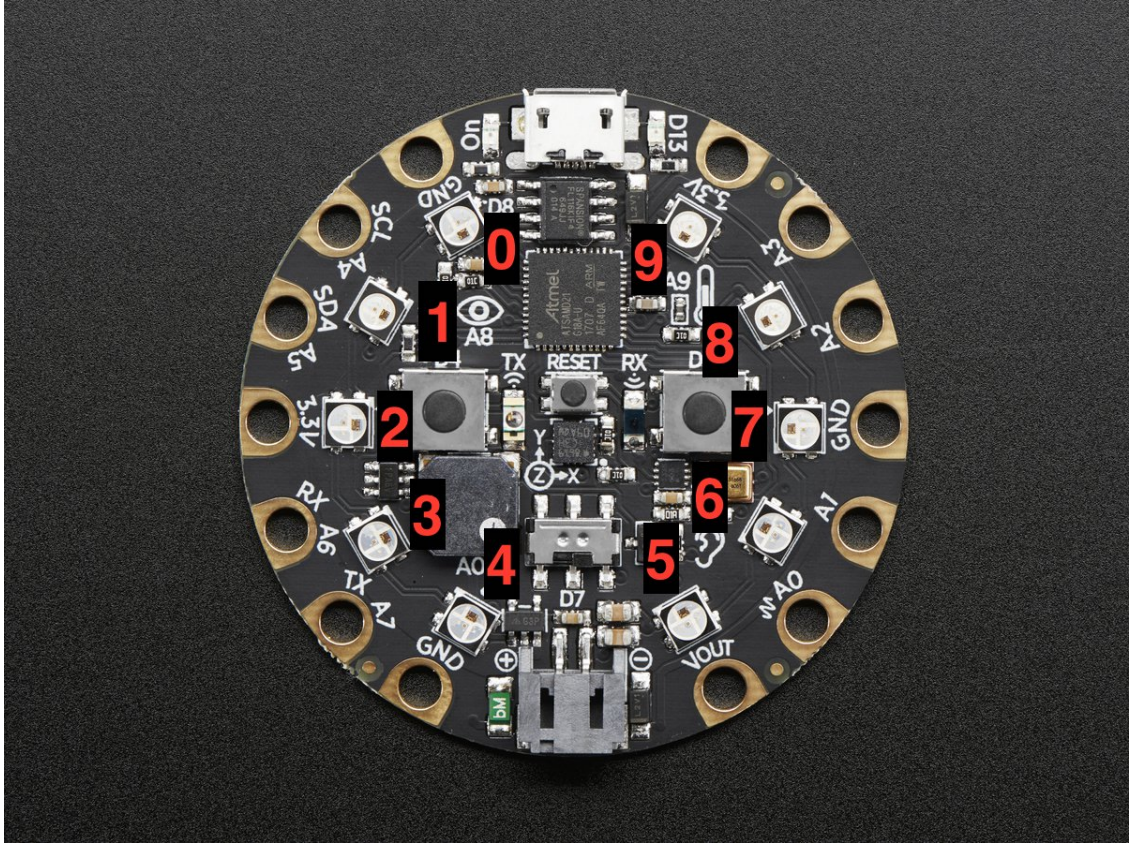
```
from adafruit_circuitplayground.express import cpx
import time

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

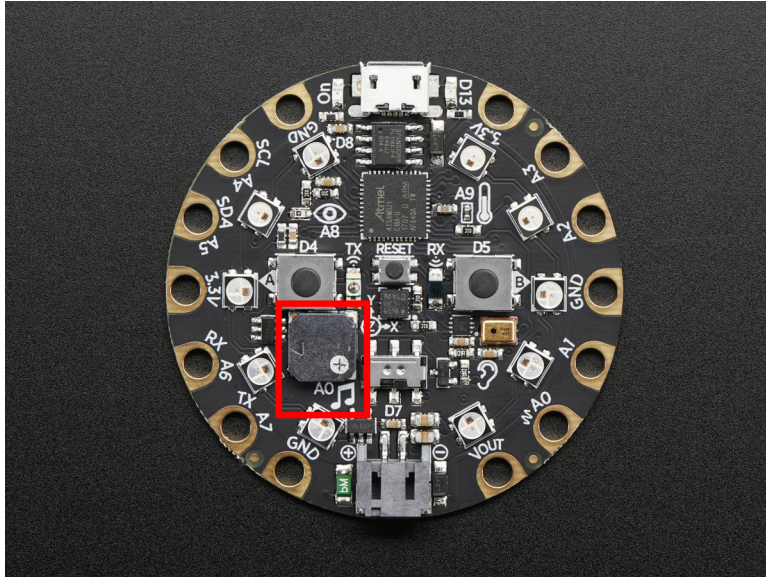
cpx.pixels.brightness = 0.3
cpx.pixels[0] = 0x003000
cpx.pixels[9] = (30, 0, 0)

# 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

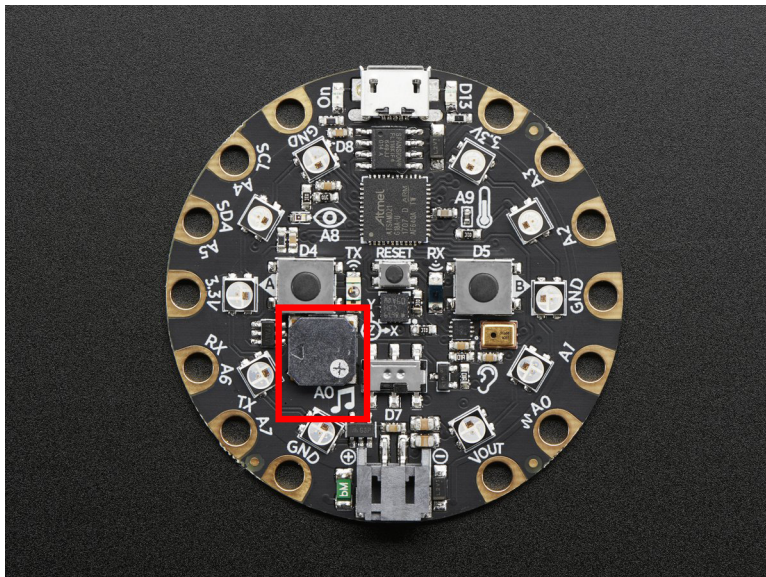
while True:
    if cpx.button_a:
        cpx.play_file("laugh.wav")
    elif cpx.button_b:
        cpx.play_file("rimshot.wav")
```

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

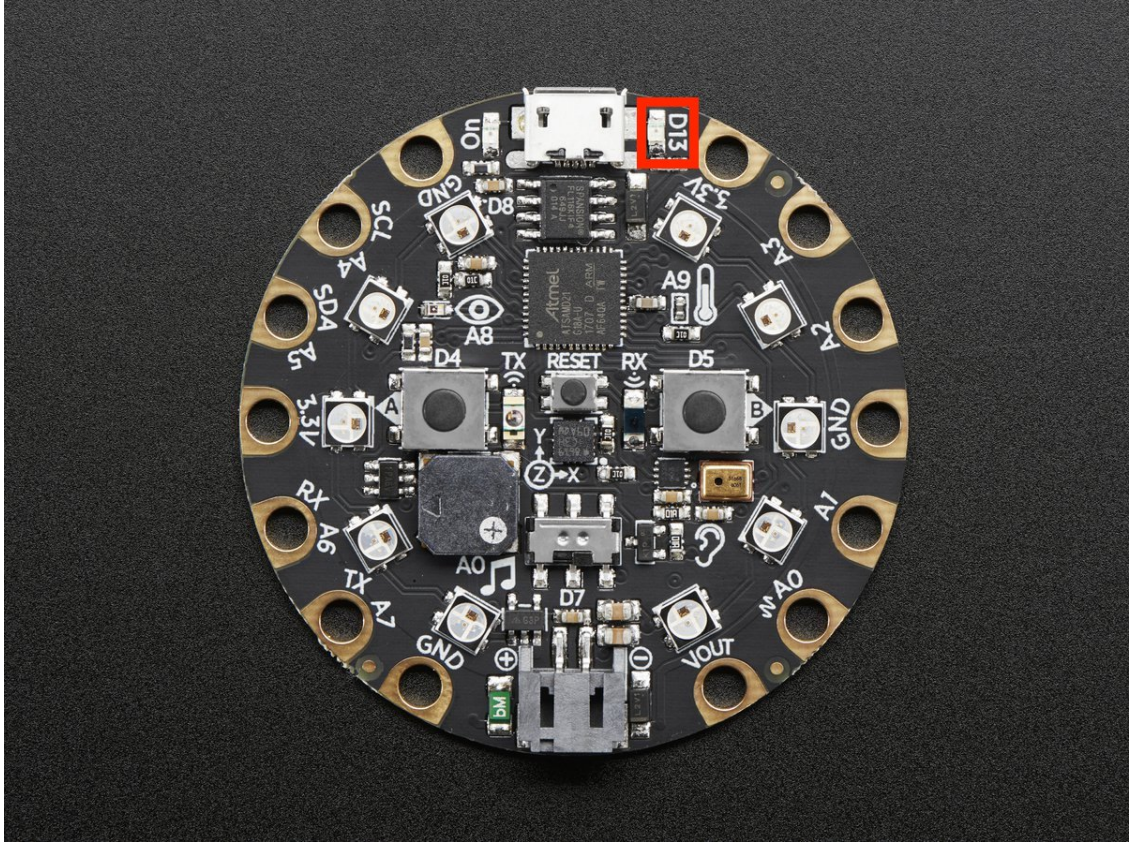


```
from adafruit_circuitplayground.express import cpx

cpx.play_tone(440, 1)
```

red_led

The red led next to the USB plug marked D13.



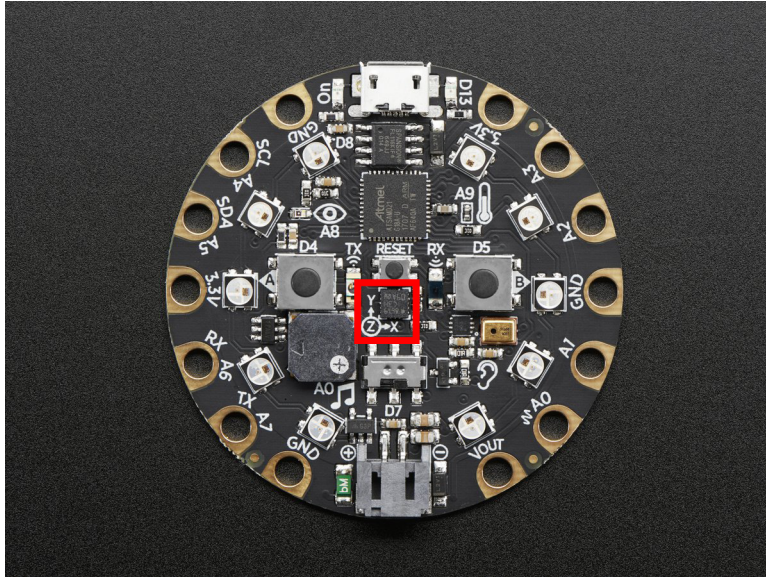
```
from adafruit_circuitplayground.express import cpx
import time

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.

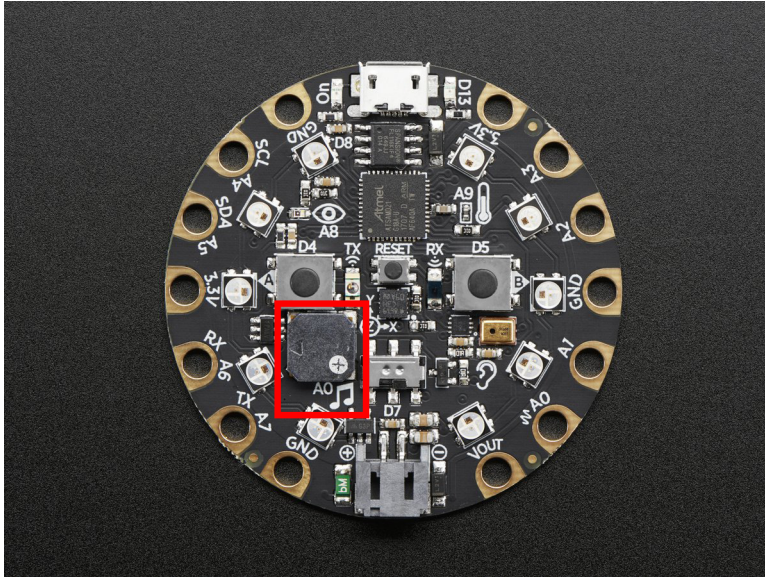
```
from adafruit_circuitplayground.express import cpx

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

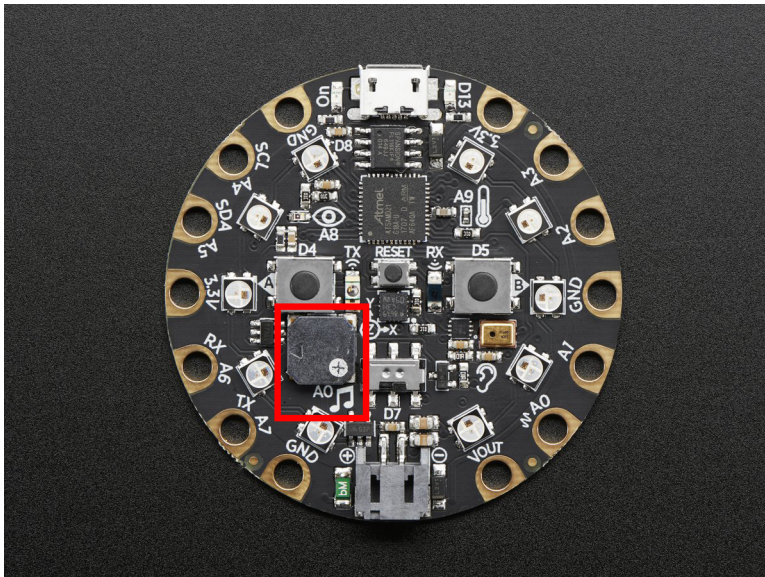


```
from adafruit_circuitplayground.express import cpx

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:
```

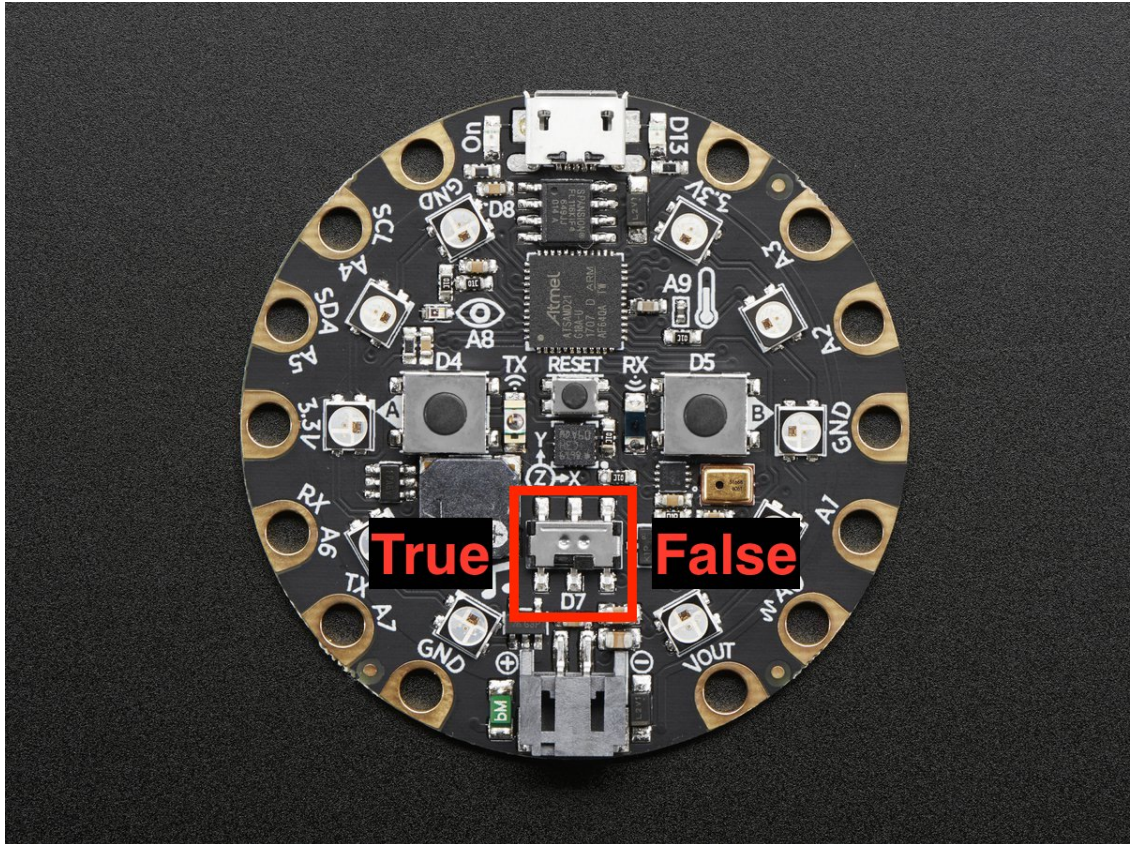
```

    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.



```

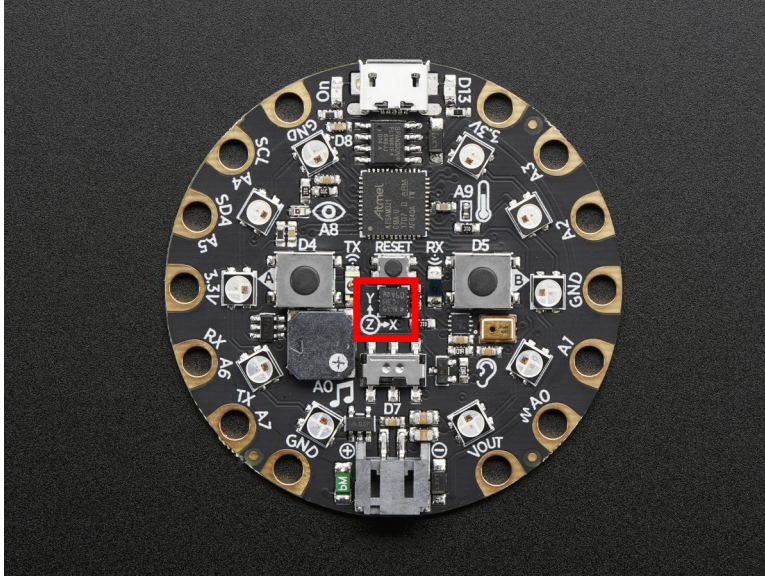
from adafruit_circuitplayground.express import cpx
import time

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

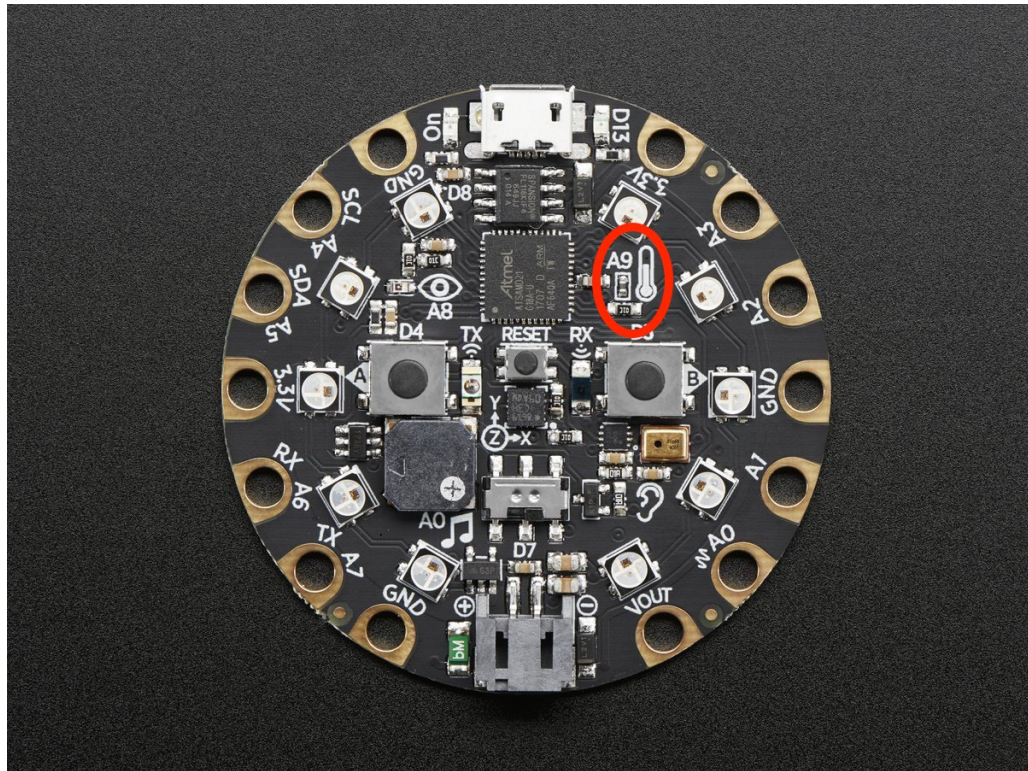
# We're looking for 2 single-taps before moving on.
while tap_count < 2:
    if cpx.tapped:
        tap_count += 1
print("Reached 2 single-taps!")

# 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.



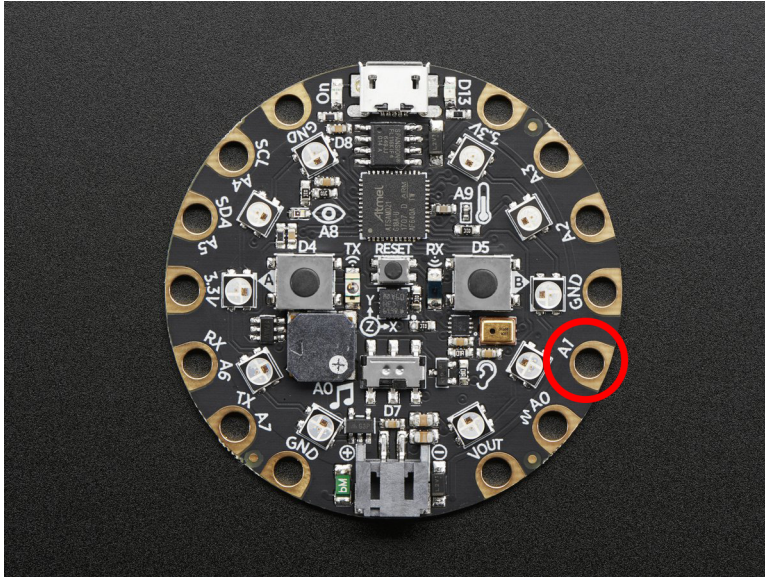
Converting this to Farenheit is easy!

```
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.

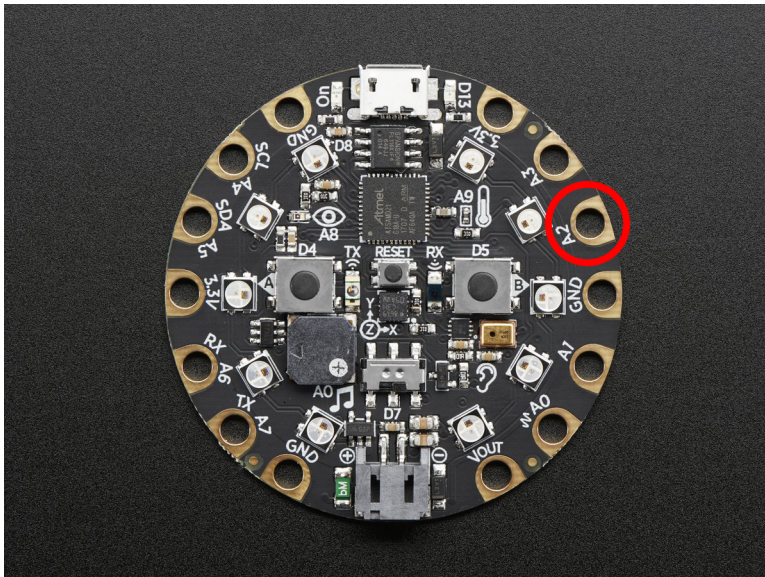


```
from adafruit_circuitplayground.express import cpx

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

`touch_A2`

Detect touch on capacitive touch pad A2.

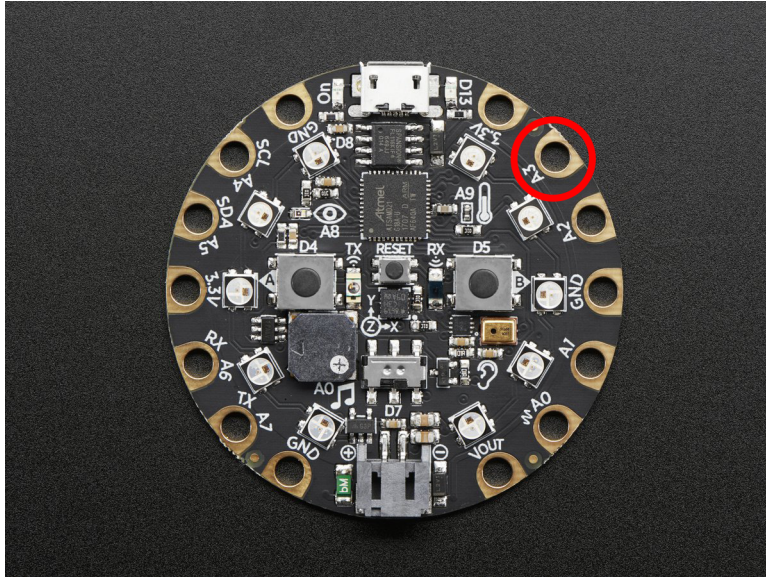


```
from adafruit_circuitplayground.express import cpx

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

`touch_A3`

Detect touch on capacitive touch pad A3.

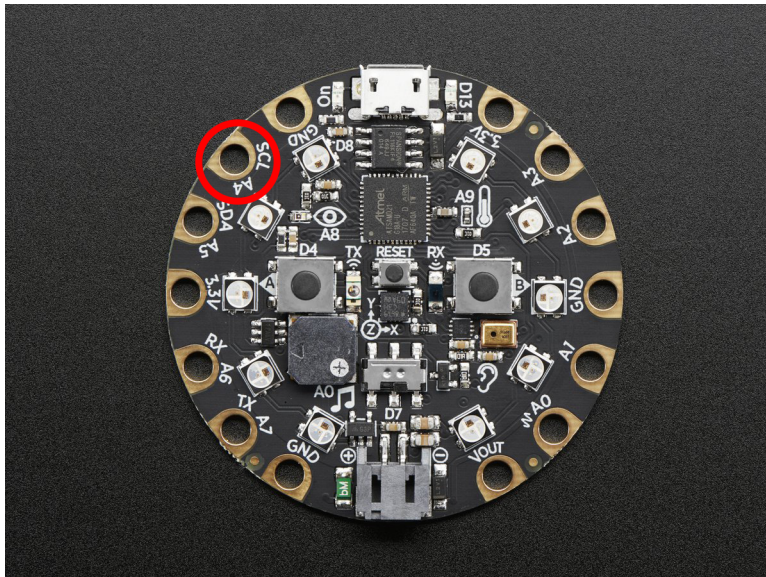


```
from adafruit_circuitplayground.express import cpx

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

touch_A4

Detect touch on capacitive touch pad A4.

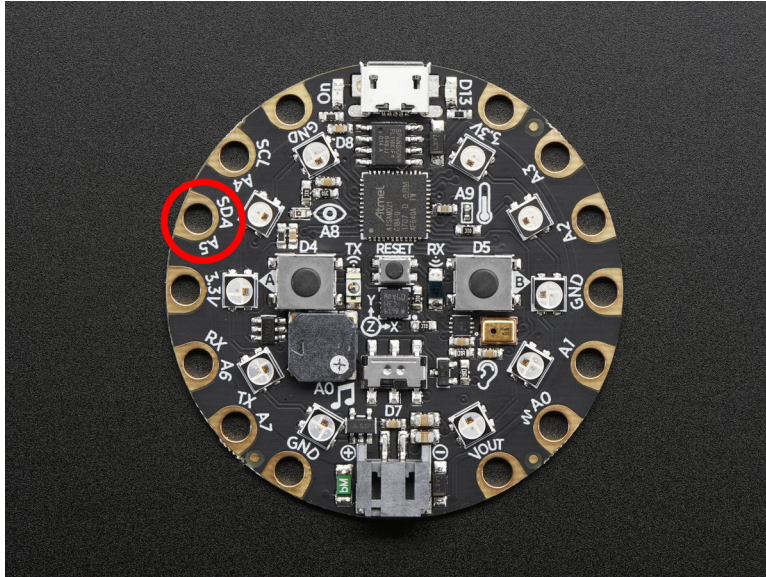


```
from adafruit_circuitplayground.express import cpx

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

touch_A5

Detect touch on capacitive touch pad A5.

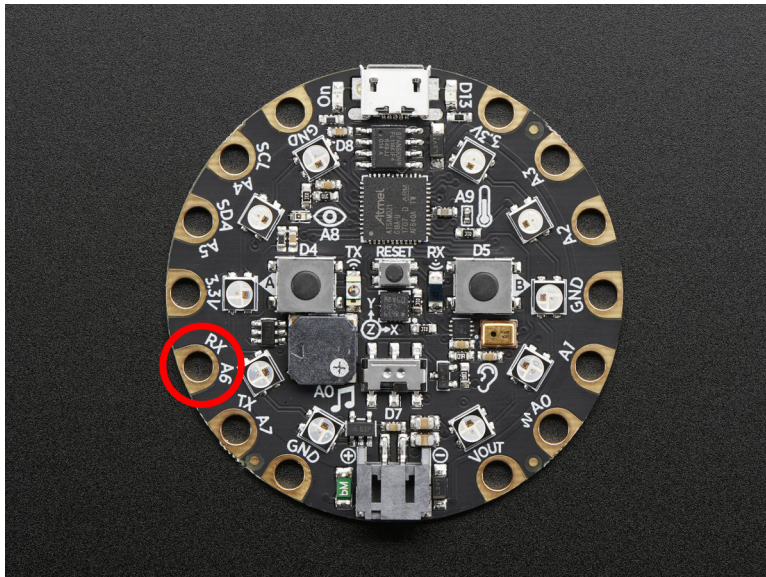


```
from adafruit_circuitplayground.express import cpx

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

touch_A6

Detect touch on capacitive touch pad A6.

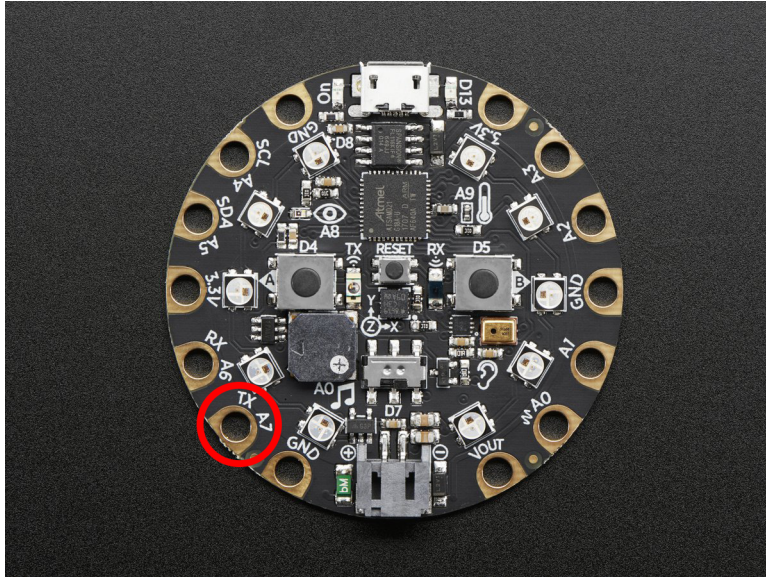


```
from adafruit_circuitplayground.express import cpx

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`, [13](#)

A

acceleration (adafruit_circuitplayground.express.Express attribute), 13
adafruit_circuitplayground.express (module), 13
adjust_touch_threshold()
 (adafruit_circuitplayground.express.Express method), 14

B

button_a (adafruit_circuitplayground.express.Express attribute), 15
button_b (adafruit_circuitplayground.express.Express attribute), 15

C

cpx (in module adafruit_circuitplayground.express), 30

D

detect_taps (adafruit_circuitplayground.express.Express attribute), 16

E

Express (class in adafruit_circuitplayground.express), 13

L

light (adafruit_circuitplayground.express.Express attribute), 17
light (adafruit_circuitplayground.express.PhotoCell attribute), 30

P

PhotoCell (class in adafruit_circuitplayground.express), 30
pixels (adafruit_circuitplayground.express.Express attribute), 18
play_file() (adafruit_circuitplayground.express.Express method), 19
play_tone() (adafruit_circuitplayground.express.Express method), 20

R

red_led (adafruit_circuitplayground.express.Express attribute), 21

S

shake() (adafruit_circuitplayground.express.Express method), 21
start_tone() (adafruit_circuitplayground.express.Express method), 22
stop_tone() (adafruit_circuitplayground.express.Express method), 23
switch (adafruit_circuitplayground.express.Express attribute), 24

T

tapped (adafruit_circuitplayground.express.Express attribute), 24
temperature (adafruit_circuitplayground.express.Express attribute), 25
touch_A1 (adafruit_circuitplayground.express.Express attribute), 26
touch_A2 (adafruit_circuitplayground.express.Express attribute), 27
touch_A3 (adafruit_circuitplayground.express.Express attribute), 27
touch_A4 (adafruit_circuitplayground.express.Express attribute), 28
touch_A5 (adafruit_circuitplayground.express.Express attribute), 28
touch_A6 (adafruit_circuitplayground.express.Express attribute), 29
touch_A7 (adafruit_circuitplayground.express.Express attribute), 29