Adafruit CircuitPlayground Library Documentation

Release 1.0

Scott Shawcroft

Dec 17, 2019

Contents

1 Installation **3**

2 Usage Example **5**

3 Contributing **7**

4 Documentation **9**

5 Table of Contents **11**

5.1 Simple test 11

5.2 `adafruit_circuitplayground.circuit_playground_base` 25

5.3 `adafruit_circuitplayground.bluefruit` 46

5.3.1 Implementation Notes 47

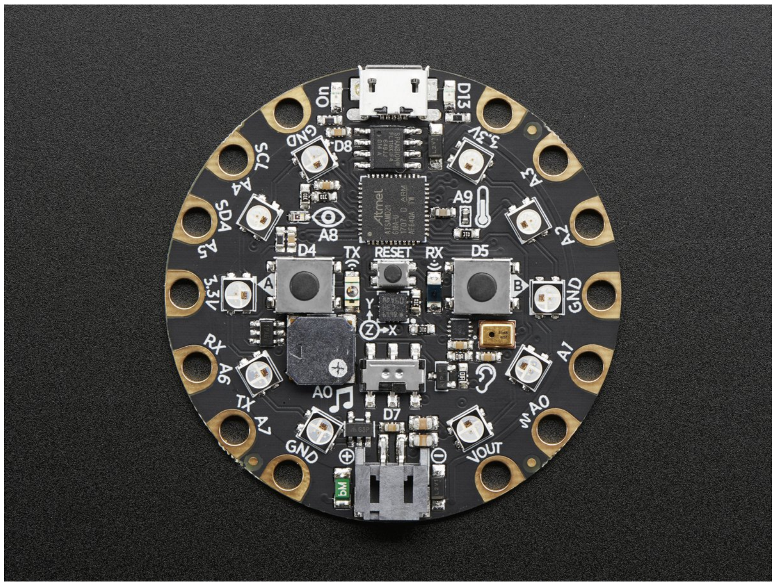
5.4 `adafruit_circuitplayground.express` 48

6 Indices and tables **51**

Python Module Index **53**

Index **55**

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

Documentation

For information on building library documentation, please check out this guide.

CHAPTER 5

Table of Contents

5.1 Simple test

Ensure your device works with this simple test.

Listing 1: examples/circuitplayground_acceleration.py

```python
"""This example uses the accelerometer on the Circuit Playground. It prints the␣
↪values. Try moving
the board to see the values change."""
import time
from adafruit_circuitplayground import cp

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

    time.sleep(0.1)
```

Listing 2: examples/circuitplayground_pixels_simpletest.py

```python
"""This example lights up the NeoPixels with a rainbow swirl."""
import time
from adafruit_circuitplayground import cp


def wheel(pos):
    # Input a value 0 to 255 to get a color value.
    # The colours are a transition r - g - b - back to r.
    if (pos < 0) or (pos > 255):
        return (0, 0, 0)
    if pos < 85:
        return (int(pos * 3), int(255 - (pos*3)), 0)
    if pos < 170:
```

(continues on next page)

(continued from previous page)

```python
        pos -= 85
        return (int(255 - pos*3), 0, int(pos*3))
    pos -= 170
    return (0, int(pos*3), int(255 - pos*3))


def rainbow_cycle(wait):
    for j in range(255):
        for i in range(cp.pixels.n):
            idx = int((i * 256 / len(cp.pixels)) + j)
            cp.pixels[i] = wheel(idx & 255)
        cp.pixels.show()
        time.sleep(wait)


cp.pixels.auto_write = False
cp.pixels.brightness = 0.3
while True:
    rainbow_cycle(0.001)    # rainbowcycle with 1ms delay per step
```

Listing 3: examples/circuitplayground_shake.py

```python
"""This example prints to the serial console when the Circuit Playground is shaken."""
from adafruit_circuitplayground import cp

while True:
    if cp.shake(shake_threshold=20):
        print("Shake detected!")
```

Listing 4: examples/circuitplayground_tapdetect_single_double.py

```python
"""This example shows how you can use single-tap and double-tap together with a delay␣
↪between.
Single-tap the board twice and then double-tap the board twice to complete the␣
↪program."""
from adafruit_circuitplayground import cp

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

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

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

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

(continues on next page)

(continued from previous page)

```python
print("Done.")
```

Listing 5: examples/circuitplayground_tapdetect.py

```python
"""This example prints to the serial console when the board is tapped."""
from adafruit_circuitplayground import cp

cp.detect_taps = 1

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

Listing 6: examples/circuitplayground_tone.py

```python
"""This example plays a different tone for each button, while the button is pressed.""
↪"
from adafruit_circuitplayground import cp

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

Listing 7: examples/circuitplayground_touched.py

```python
"""This example prints to the serial console when you touch the capacitive touch pads.
↪"""
from adafruit_circuitplayground import cp

while True:
    if cp.touch_A1:
        print('Touched pad A1')
    if cp.touch_A2:
        print('Touched pad A2')
    if cp.touch_A3:
        print('Touched pad A3')
    if cp.touch_A4:
        print('Touched pad A4')
    if cp.touch_A5:
        print('Touched pad A5')
    if cp.touch_A6:
        print('Touched pad A6')
    if cp.touch_TX:
        print('Touched pad TX')
```

Listing 8: examples/circuitplayground_acceleration_neopixels.py

```python
"""If the switch is to the right, it will appear that nothing is happening. Move the␣
↪switch to the
left to see the NeoPixels light up in colors related to the accelerometer! The␣
↪Circuit Playground
```

(continues on next page)

(continued from previous page)

```python
has an accelerometer in the center that returns (x, y, z) acceleration values. This␣
↪program uses
those values to light up the NeoPixels based on those acceleration values."""
from adafruit_circuitplayground import cp

# 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 not cp.switch:
        # If the switch is to the right, it returns False!
        print("Slide switch off!")
        cp.pixels.fill((0, 0, 0))
        continue
    else:
        R = 0
        G = 0
        B = 0
        x, y, z = cp.acceleration
        print((x, y, z))
        cp.pixels.fill(((R + abs(int(x))), (G + abs(int(y))), (B + abs(int(z)))))
```

Listing 9: examples/circuitplayground_button_a.py

```python
"""This example turns on the little red LED when button A is pressed."""
from adafruit_circuitplayground import cp

while True:
    if cp.button_a:
        print("Button A pressed!")
        cp.red_led = True
```

Listing 10: examples/circuitplayground_button_b.py

```python
"""This example turns the little red LED on only while button B is currently being␣
↪pressed."""
from adafruit_circuitplayground import cp

# This code is written to be readable versus being Pylint compliant.
# pylint: disable=simplifiable-if-statement

while True:
    if cp.button_b:
        cp.red_led = True
    else:
        cp.red_led = False

# Can also be written as:
#    cp.red_led = cp.button_b
```

Listing 11: examples/circuitplayground_buttons_1_neopixel.py

```python
"""This example lights up the third NeoPixel while button A is being pressed, and␣
↪lights up the
eighth NeoPixel while button B is being pressed."""
from adafruit_circuitplayground import cp
```

(continues on next page)

(continued from previous page)

```

cp.pixels.brightness = 0.3
cp.pixels.fill((0, 0, 0))  # Turn off the NeoPixels if they're on!

while True:
    if cp.button_a:
        cp.pixels[2] = (0, 255, 0)
    else:
        cp.pixels[2] = (0, 0, 0)

    if cp.button_b:
        cp.pixels[7] = (0, 0, 255)
    else:
        cp.pixels[7] = (0, 0, 0)
```

Listing 12: examples/circuitplayground_buttons_neopixels.py

```
"""This example lights up half the NeoPixels red while button A is being pressed, and␣
↪half the
NeoPixels green while button B is being pressed."""
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.3
cp.pixels.fill((0, 0, 0))  # Turn off the NeoPixels if they're on!

while True:
    if cp.button_a:
        cp.pixels[0:5] = [(255, 0, 0)] * 5
    else:
        cp.pixels[0:5] = [(0, 0, 0)] * 5

    if cp.button_b:
        cp.pixels[5:10] = [(0, 255, 0)] * 5
    else:
        cp.pixels[5:10] = [(0, 0, 0)] * 5
```

Listing 13: examples/circuitplayground_ir_receive.py

```
"""THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
This example requires the adafruit_irremote.mpy library.

THIS EXAMPLE WORKS WITH CIRCUIT PLAYGROUND EXPRESS ONLY.

This example uses the IR receiver found near the center of the board. Works with␣
↪another Circuit
Playground Express running the circuitplayground_ir_transmit.py example. The␣
↪NeoPixels will light
up when the buttons on the TRANSMITTING Circuit Playground Express are pressed!"""
import pulseio
import board
import adafruit_irremote
from adafruit_circuitplayground import cp

# Create a 'pulseio' input, to listen to infrared signals on the IR receiver
try:
    pulsein = pulseio.PulseIn(board.IR_RX, maxlen=120, idle_state=True)
```

(continues on next page)

(continued from previous page)

```python
except AttributeError:
    raise NotImplementedError("This example does not work with Circuit Playground␣
↪Bluefruti!")
# Create a decoder that will take pulses and turn them into numbers
decoder = adafruit_irremote.GenericDecode()

while True:
    cp.red_led = True
    pulses = decoder.read_pulses(pulsein)
    try:
        # Attempt to convert received pulses into numbers
        received_code = decoder.decode_bits(pulses)
    except adafruit_irremote.IRNECRepeatException:
        # We got an unusual short code, probably a 'repeat' signal
        continue
    except adafruit_irremote.IRDecodeException:
        # Something got distorted
        continue

    print("Infrared code received: ", received_code)
    if received_code == [66, 84, 78, 65]:
        print("Button A signal")
        cp.pixels.fill((100, 0, 155))
    if received_code == [66, 84, 78, 64]:
        print("Button B Signal")
        cp.pixels.fill((210, 45, 0))
```

Listing 14: examples/circuitplayground_ir_transmit.py

```python
"""THIS EXAMPLE REQUIRES A SEPARATE LIBRARY BE LOADED ONTO YOUR CIRCUITPY DRIVE.
This example requires the adafruit_irremote.mpy library.

THIS EXAMPLE WORKS WITH CIRCUIT PLAYGROUND EXPRESS ONLY.

This example uses the IR transmitter found near the center of the board. Works with␣
↪another Circuit
Playground Express running the circuitplayground_ir_receive.py example. Press the␣
↪buttons to light
up the NeoPixels on the RECEIVING Circuit Playground Express!"""
import time
import pulseio
import board
import adafruit_irremote
from adafruit_circuitplayground import cp

# Create a 'pulseio' output, to send infrared signals from the IR transmitter
try:
    pwm = pulseio.PWMOut(board.IR_TX, frequency=38000, duty_cycle=2 ** 15)
except AttributeError:
    raise NotImplementedError("This example does not work with Circuit Playground␣
↪Bluefruit!")
pulseout = pulseio.PulseOut(pwm)
# Create an encoder that will take numbers and turn them into NEC IR pulses
encoder = adafruit_irremote.GenericTransmit(header=[9500, 4500], one=[550, 550],
                                            zero=[550, 1700], trail=0)

```

(continues on next page)

(continued from previous page)

```python
while True:
    if cp.button_a:
        print("Button A pressed! \n")
        cp.red_led = True
        encoder.transmit(pulseout, [66, 84, 78, 65])
        cp.red_led = False
        # wait so the receiver can get the full message
        time.sleep(0.2)
    if cp.button_b:
        print("Button B pressed! \n")
        cp.red_led = True
        encoder.transmit(pulseout, [66, 84, 78, 64])
        cp.red_led = False
        time.sleep(0.2)
```

Listing 15: examples/circuitplayground_light_neopixels.py

```python
"""
This example uses the light sensor on the Circuit Playground, located next to the␣
↪picture of the
eye on the board. Once you have the library loaded, try shining a flashlight on your␣
↪Circuit
Playground to watch the number of NeoPixels lit up increase, or try covering up the␣
↪light sensor
to watch the number decrease.
"""

import time
from adafruit_circuitplayground import cp

cp.pixels.auto_write = False
cp.pixels.brightness = 0.3


def scale_range(value):
    """Scale a value from 0-320 (light range) to 0-10 (the number of NeoPixels).
    Allows remapping light value to pixel position."""
    return int(value / 320 * 10)


while True:
    peak = scale_range(cp.light)
    print(cp.light)
    print(int(peak))

    for i in range(10):
        if i <= peak:
            cp.pixels[i] = (0, 255, 255)
        else:
            cp.pixels[i] = (0, 0, 0)
    cp.pixels.show()
    time.sleep(0.05)
```

Listing 16: examples/circuitplayground_light.py

```python
"""This example uses the light sensor on your Circuit Playground, located next to the␣
↪picture of
the eye. Try shining a flashlight on your Circuit Playground, or covering the light␣
↪sensor with
your finger to see the values increase and decrease."""
import time
from adafruit_circuitplayground import cp

while True:
    print("Light:", cp.light)
    time.sleep(0.2)
```

Listing 17: examples/circuitplayground_neopixel_0_1.py

```python
"""This example lights up the first and second NeoPixel, red and blue respectively."""
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.3

while True:
    cp.pixels[0] = (255, 0, 0)
    cp.pixels[1] = (0, 0, 255)
```

Listing 18: examples/circuitplayground_light_plotter.py

```python
"""If you're using Mu, this example will plot the light levels from the light sensor␣
↪(located next
to the eye) on your Circuit Playground. Try shining a flashlight on your Circuit␣
↪Playground, or
covering the light sensor to see the plot increase and decrease."""
import time
from adafruit_circuitplayground import cp

while True:
    print("Light:", cp.light)
    print((cp.light,))
    time.sleep(0.1)
```

Listing 19: examples/circuitplayground_play_file_buttons.py

```python
"""THIS EXAMPLE REQUIRES A WAV FILE FROM THE examples FOLDER IN THE
Adafruit_CircuitPython_CircuitPlayground REPO found at:
https://github.com/adafruit/Adafruit_CircuitPython_CircuitPlayground/tree/master/
↪examples

Copy the "dip.wav" and "rise.wav" files to your CIRCUITPY drive.

Once the files are copied, this example plays a different wav file for each button␣
↪pressed!"""
from adafruit_circuitplayground import cp

while True:
    if cp.button_a:
        cp.play_file("dip.wav")
```

(continues on next page)

(continued from previous page)

```
    if cp.button_b:
        cp.play_file("rise.wav")
```

Listing 20: examples/circuitplayground_play_file.py

```
"""THIS EXAMPLE REQUIRES A WAV FILE FROM THE examples FOLDER IN THE
Adafruit_CircuitPython_CircuitPlayground REPO found at:
https://github.com/adafruit/Adafruit_CircuitPython_CircuitPlayground/tree/master/
↪examples

Copy the "dip.wav" file to your CIRCUITPY drive.

Once the file is copied, this example plays a wav file!"""
from adafruit_circuitplayground import cp

cp.play_file("dip.wav")
```

Listing 21: examples/circuitplayground_play_tone_buttons.py

```
"""This example plays a different tone for a duration of 1 second for each button␣
↪pressed."""
from adafruit_circuitplayground import cp

while True:
    if cp.button_a:
        cp.play_tone(262, 1)
    if cp.button_b:
        cp.play_tone(294, 1)
```

Listing 22: examples/circuitplayground_play_tone.py

```
"""This example plays two tones for 1 second each. Note that the tones are not in a␣
↪loop - this is
to prevent them from playing indefinitely!"""
from adafruit_circuitplayground import cp

cp.play_tone(262, 1)
cp.play_tone(294, 1)
```

Listing 23: examples/circuitplayground_red_led_blinky.py

```
"""This is the "Hello, world!" of CircuitPython: Blinky! This example blinks the␣
↪little red LED on
and off!"""
import time
from adafruit_circuitplayground import cp

while True:
    cp.red_led = True
    time.sleep(0.5)
    cp.red_led = False
    time.sleep(0.5)
```

Listing 24: examples/circuitplayground_red_led_blnky_short.py

```python
"""This is the "Hello, world!" of CircuitPython: Blinky! This example blinks the␣
↪little red LED on
and off! It's a shorter version of the other Blinky example."""
import time
from adafruit_circuitplayground import cp

while True:
    cp.red_led = not cp.red_led
    time.sleep(0.5)
```

Listing 25: examples/circuitplayground_red_led.py

```python
"""This example turns on the little red LED."""
from adafruit_circuitplayground import cp

while True:
    cp.red_led = True
```

Listing 26: examples/circuitplayground_slide_switch_red_led.py

```python
"""This example uses the slide switch to control the little red LED."""
from adafruit_circuitplayground import cp

# This code is written to be readable versus being Pylint compliant.
# pylint: disable=simplifiable-if-statement

while True:
    if cp.switch:
        cp.red_led = True
    else:
        cp.red_led = False
```

Listing 27: examples/circuitplayground_slide_switch_red_led_short.py

```python
"""This example uses the slide switch to control the little red LED. When the switch␣
↪is to the
right it returns False, and when it's to the left, it returns True."""
from adafruit_circuitplayground import cp

while True:
    cp.red_led = cp.switch
```

Listing 28: examples/circuitplayground_slide_switch.py

```python
"""This example prints the status of the slide switch. Try moving the switch back and␣
↪forth to see
what's printed to the serial console!"""
import time
from adafruit_circuitplayground import cp

while True:
    print("Slide switch:", cp.switch)
    time.sleep(0.1)
```

Listing 29: examples/circuitplayground_sound_meter.py

```python
"""This example uses the sound sensor, located next to the picture of the ear on your board, to
light up the NeoPixels as a sound meter. Try talking to your Circuit Playground or clapping, etc,
to see the NeoPixels light up!"""
import array
import math
import audiobusio
import board
from adafruit_circuitplayground import cp


def constrain(value, floor, ceiling):
    return max(floor, min(value, ceiling))


def log_scale(input_value, input_min, input_max, output_min, output_max):
    normalized_input_value = (input_value - input_min) / (input_max - input_min)
    return output_min + math.pow(normalized_input_value, 0.630957) * (output_max - output_min)


def normalized_rms(values):
    minbuf = int(sum(values) / len(values))
    return math.sqrt(sum(float(sample - minbuf) *
                         (sample - minbuf) for sample in values) / len(values))


mic = audiobusio.PDMIn(board.MICROPHONE_CLOCK, board.MICROPHONE_DATA,
                       sample_rate=16000, bit_depth=16)

samples = array.array('H', [0] * 160)
mic.record(samples, len(samples))
input_floor = normalized_rms(samples) + 10

# Lower number means more sensitive - more LEDs will light up with less sound.
sensitivity = 500
input_ceiling = input_floor + sensitivity

peak = 0
while True:
    mic.record(samples, len(samples))
    magnitude = normalized_rms(samples)
    print((magnitude,))

    c = log_scale(constrain(magnitude, input_floor, input_ceiling),
                  input_floor, input_ceiling, 0, 10)

    cp.pixels.fill((0, 0, 0))
    for i in range(10):
        if i < c:
            cp.pixels[i] = (i * (255 // 10), 50, 0)
        if c >= peak:
            peak = min(c, 10 - 1)
        elif peak > 0:
```

(continues on next page)

(continued from previous page)

```
            peak = peak - 1
        if peak > 0:
            cp.pixels[int(peak)] = (80, 0, 255)
    cp.pixels.show()
```

Listing 30: examples/circuitplayground_tap_red_led.py

```
"""This example turns on the little red LED and prints to the serial console when you␣
↪double-tap
the Circuit Playground!"""
import time
from adafruit_circuitplayground import cp

# Change to 1 for detecting a single-tap!
cp.detect_taps = 2

while True:
    if cp.tapped:
        print("Tapped!")
        cp.red_led = True
        time.sleep(0.1)
    else:
        cp.red_led = False
```

Listing 31: examples/circuitplayground_temperature_neopixels.py

```
"""
This example use the temperature sensor on the Circuit Playground, located next to␣
↪the picture of
the thermometer on the board. Try warming up the board to watch the number of␣
↪NeoPixels lit up
increase, or cooling it down to see the number decrease. You can set the min and max␣
↪temperatures
to make it more or less sensitive to temperature changes.
"""
import time
from adafruit_circuitplayground import cp

cp.pixels.auto_write = False
cp.pixels.brightness = 0.3

# Set these based on your ambient temperature in Celsius for best results!
minimum_temp = 24
maximum_temp = 30


def scale_range(value):
    """Scale a value from the range of minimum_temp to maximum_temp (temperature␣
↪range) to 0-10
    (the number of NeoPixels). Allows remapping temperature value to pixel position.""
↪"
    return int((value - minimum_temp) / (maximum_temp - minimum_temp) * 10)


while True:
    peak = scale_range(cp.temperature)
```

(continues on next page)

(continued from previous page)

```python
    print(cp.temperature)
    print(int(peak))

    for i in range(10):
        if i <= peak:
            cp.pixels[i] = (0, 255, 255)
        else:
            cp.pixels[i] = (0, 0, 0)
    cp.pixels.show()
    time.sleep(0.05)
```

Listing 32: examples/circuitplayground_temperature_plotter.py

```python
"""If you're using Mu, this example will plot the temperature in C and F on the␣
↪plotter! Click
"Plotter" to open it, and place your finger over the sensor to see the numbers change.
↪ The
sensor is located next to the picture of the thermometer on the CPX."""
import time
from adafruit_circuitplayground import cp

while True:
    print("Temperature C:", cp.temperature)
    print("Temperature F:", cp.temperature * 1.8 + 32)
    print((cp.temperature, cp.temperature * 1.8 + 32))
    time.sleep(0.1)
```

Listing 33: examples/circuitplayground_temperature.py

```python
"""This example uses the temperature sensor on the Circuit Playground, located next␣
↪to the image of
a thermometer on the board. It prints the temperature in both C and F to the serial␣
↪console. Try
putting your finger over the sensor to see the numbers change!"""
import time
from adafruit_circuitplayground import cp

while True:
    print("Temperature C:", cp.temperature)
    print("Temperature F:", cp.temperature * 1.8 + 32)
    time.sleep(1)
```

Listing 34: examples/circuitplayground_touch_pixel_fill_rainbow.py

```python
"""This example uses the capacitive touch pads on the Circuit Playground. They are␣
↪located around
the outer edge of the board and are labeled A1-A6 and TX. (A0 is not a touch pad.)␣
↪This example
lights up all the NeoPixels a different color of the rainbow for each pad touched!"""
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.3

while True:
```

(continues on next page)

(continued from previous page)

```
    if cp.touch_A1:
        print("Touched A1!")
        cp.pixels.fill((255, 0, 0))
    if cp.touch_A2:
        print("Touched A2!")
        cp.pixels.fill((210, 45, 0))
    if cp.touch_A3:
        print("Touched A3!")
        cp.pixels.fill((155, 100, 0))
    if cp.touch_A4:
        print("Touched A4!")
        cp.pixels.fill((0, 255, 0))
    if cp.touch_A5:
        print("Touched A5!")
        cp.pixels.fill((0, 135, 125))
    if cp.touch_A6:
        print("Touched A6!")
        cp.pixels.fill((0, 0, 255))
    if cp.touch_TX:
        print("Touched TX!")
        cp.pixels.fill((100, 0, 155))
    time.sleep(0.1)
```

Listing 35: examples/circuitplayground_touch_pixel_rainbow.py

```
"""This example uses the capacitive touch pads on the Circuit Playground. They are␣
↪located around
the outer edge of the board and are labeled A1-A6 and TX. (A0 is not a touch pad.)␣
↪This example
lights up the nearest NeoPixel to that pad a different color of the rainbow!"""
import time
from adafruit_circuitplayground import cp

cp.pixels.brightness = 0.3

while True:
    if cp.touch_A1:
        print("Touched A1!")
        cp.pixels[6] = (255, 0, 0)
    if cp.touch_A2:
        print("Touched A2!")
        cp.pixels[8] = (210, 45, 0)
    if cp.touch_A3:
        print("Touched A3!")
        cp.pixels[9] = (155, 100, 0)
    if cp.touch_A4:
        print("Touched A4!")
        cp.pixels[0] = (0, 255, 0)
    if cp.touch_A5:
        print("Touched A5!")
        cp.pixels[1] = (0, 135, 125)
    if cp.touch_A6:
        print("Touched A6!")
        cp.pixels[3] = (0, 0, 255)
    if cp.touch_TX:
        print("Touched TX!")
```

(continues on next page)

(continued from previous page)

```
        cp.pixels[4] = (100, 0, 155)
    time.sleep(0.1)
```

5.2 adafruit_circuitplayground.circuit_playground_base

CircuitPython base class for Circuit Playground.

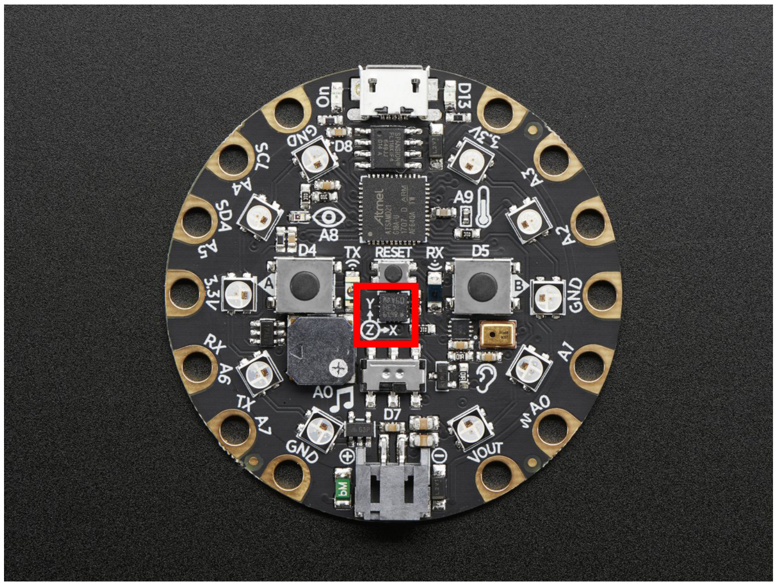
- Circuit Playground Express
- Circuit Playground Bluefruit.
- Author(s): Kattni Rembor, Scott Shawcroft

class adafruit_circuitplayground.circuit_playground_base.**CircuitPlaygroundBase**

Circuit Playground base class.

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.

To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

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

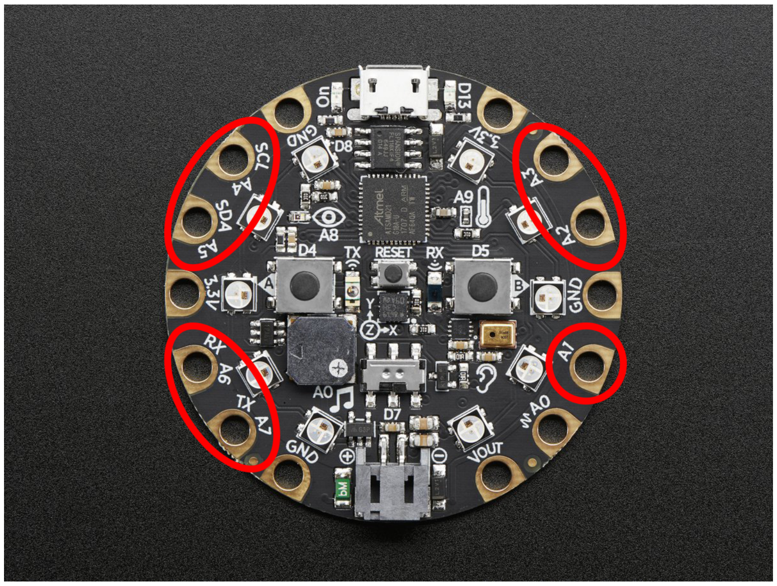
To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    x, y, z = cpb.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



To use with the Circuit Playground Express:

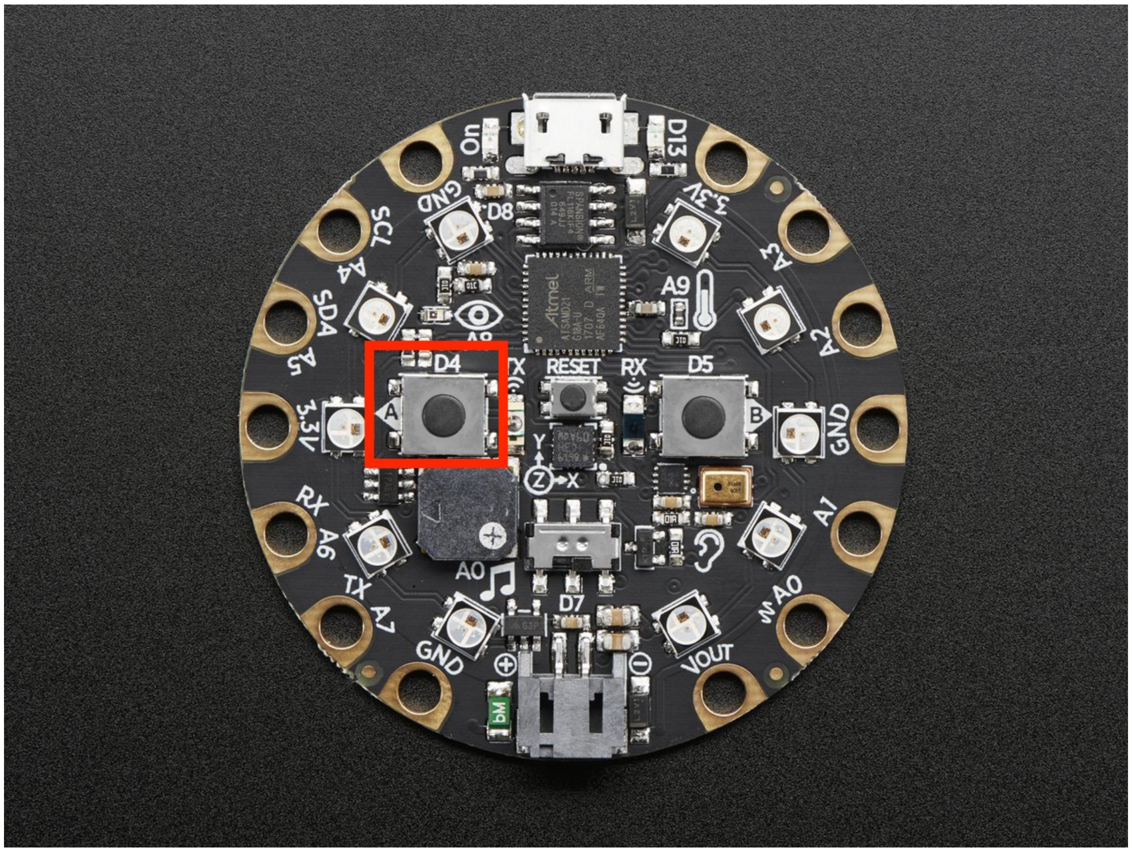
```
from adafruit_circuitplayground.express import cpx

cpx.adjust_touch_threshold(200)

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

cpb.adjust_touch_threshold(200)

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

button_a

`True` when Button A is pressed. `False` if not.



To use with the Circuit Playground Express:

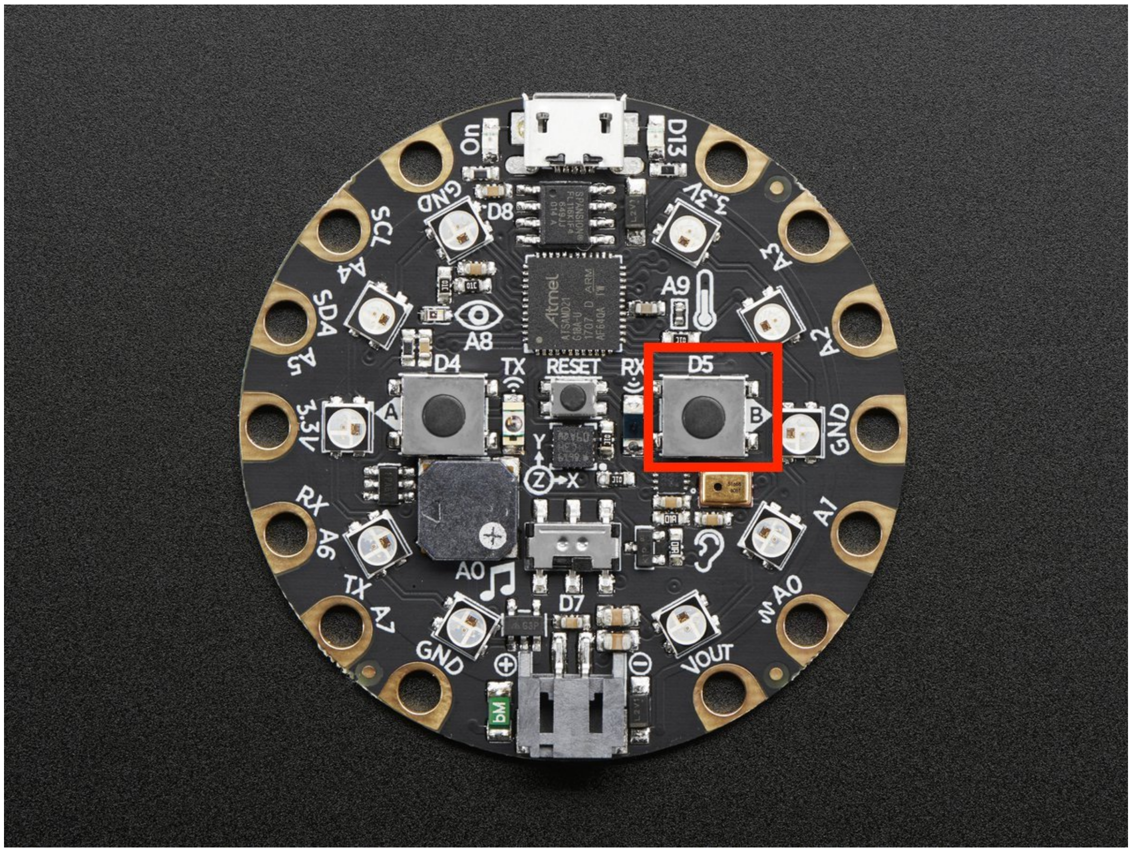
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

button_b

`True` when Button B is pressed. `False` if not.



To use with the Circuit Playground Express:

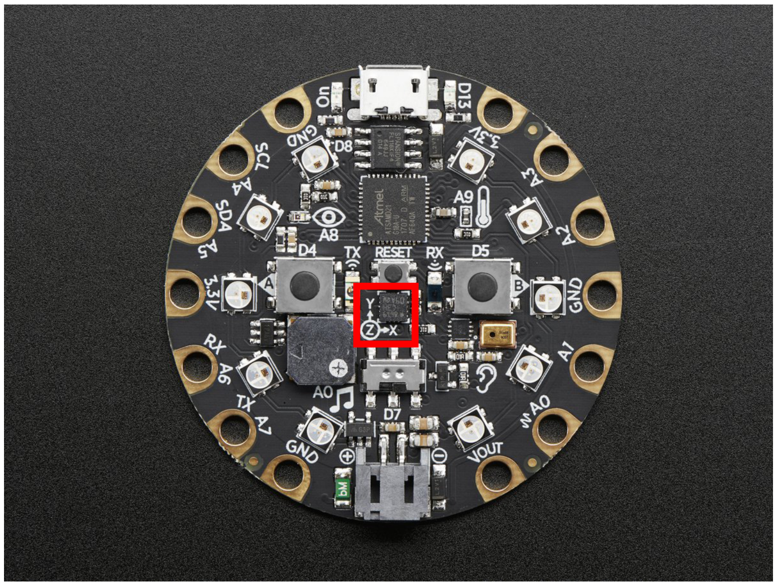
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.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`.



To use with the Circuit Playground Express:

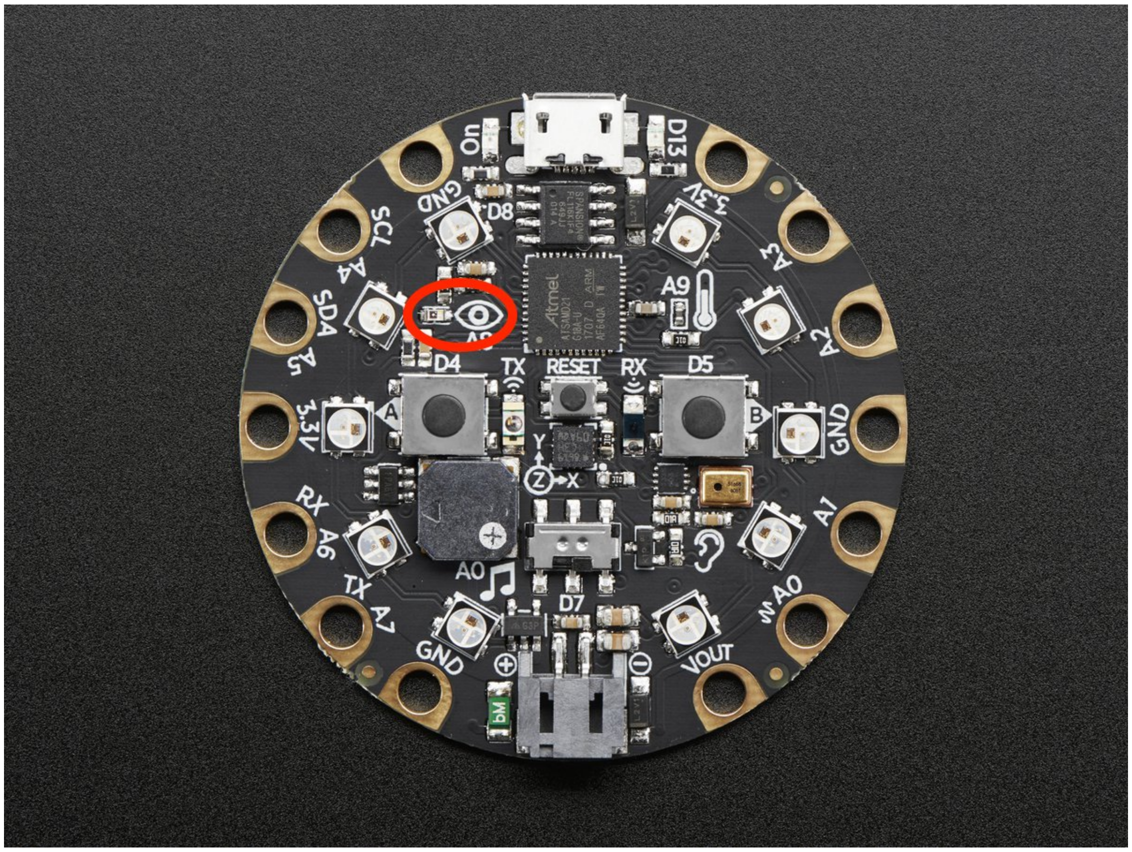
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

light

The light level.



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

To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx
import time

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

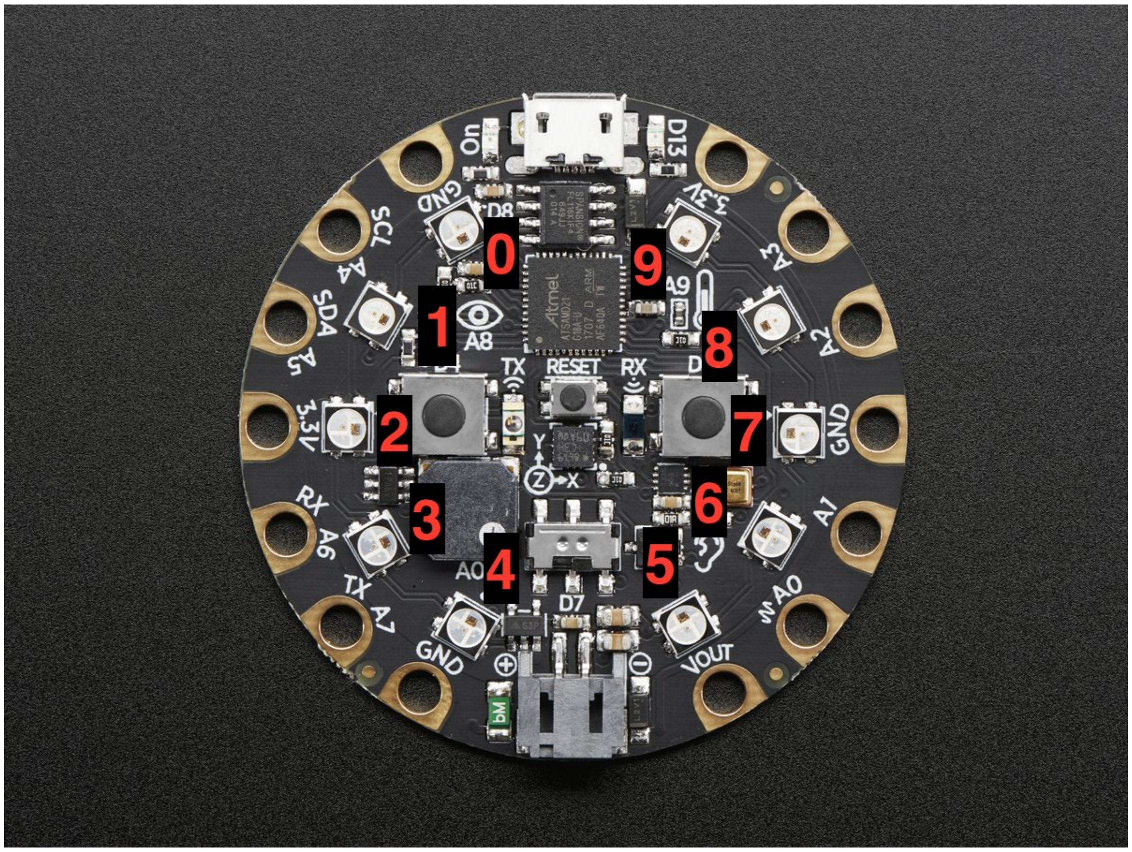
To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb
import time

while True:
    print("Light:", cpb.light)
    time.sleep(1)
```

pixels

Sequence-like object representing the ten NeoPixels around the outside of the Circuit Playground. 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 ninth red.

To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

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

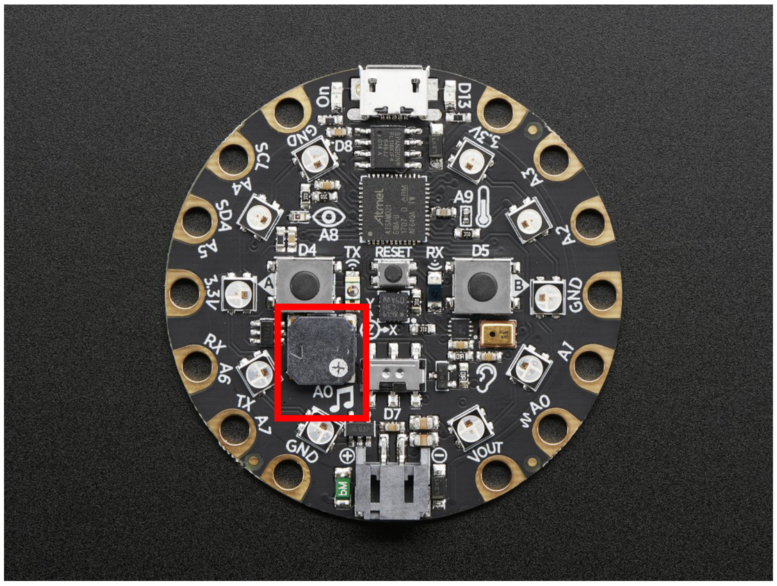
To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

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



To use with the Circuit Playground Express:

```
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")
```

To use with the Circuit Playground Bluefruit:

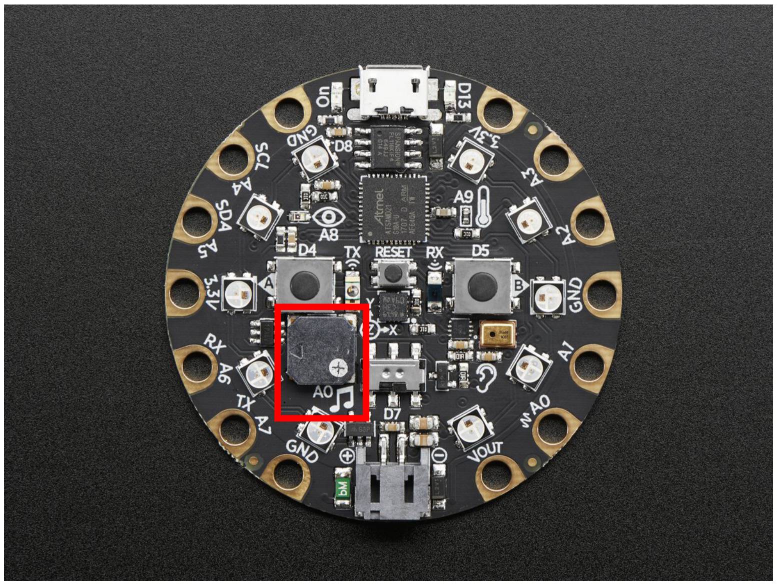
```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.button_a:
        cpb.play_file("laugh.wav")
    elif cpb.button_b:
        cpb.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



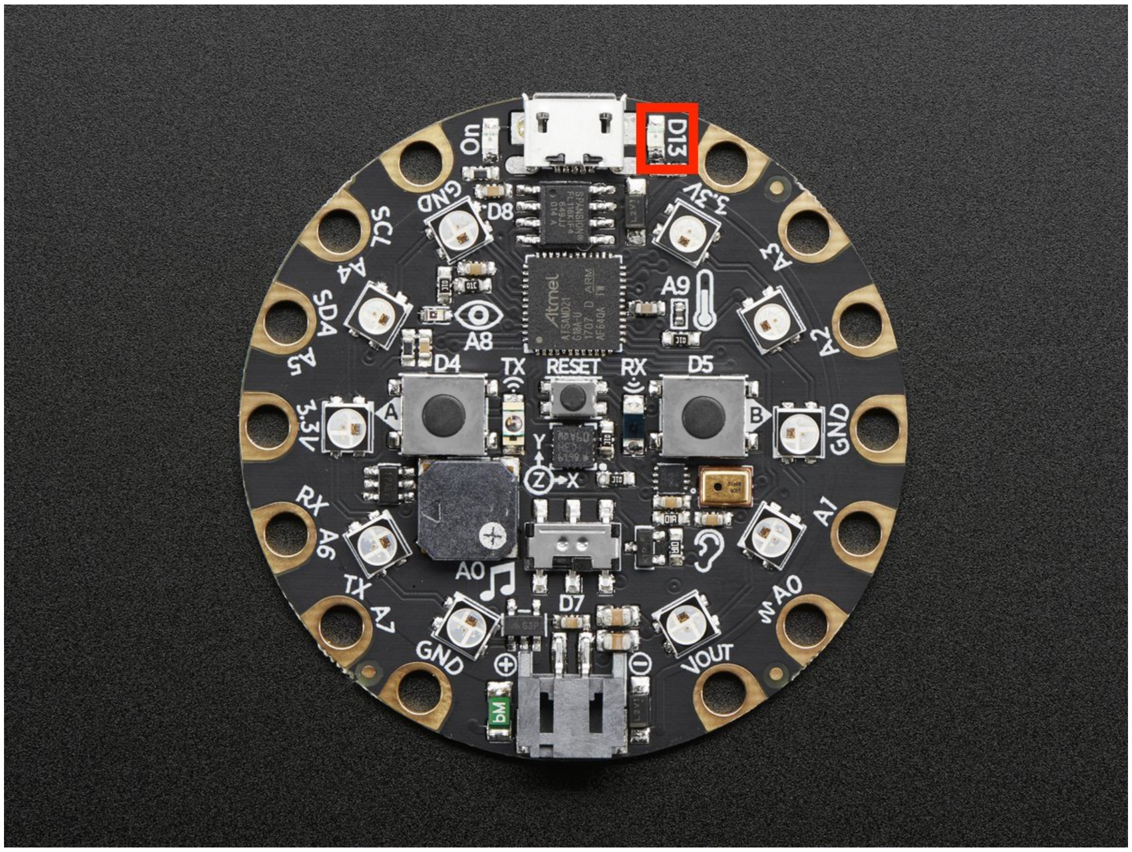
To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

cpx.play_tone(440, 1)
```

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

cpb.play_tone(440, 1)
```

red_led
The red led next to the USB plug marked D13.



To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx
import time

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

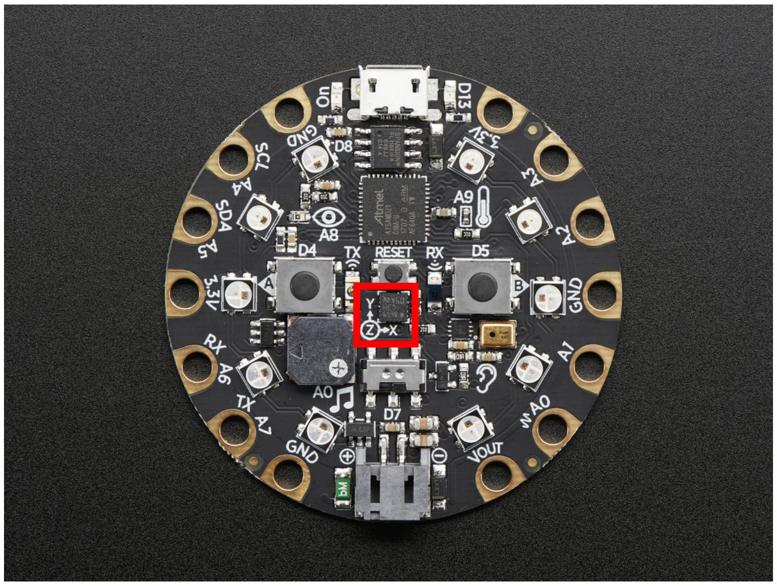
To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb
import time

while True:
    cpb.red_led = True
    time.sleep(1)
    cpb.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)



To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

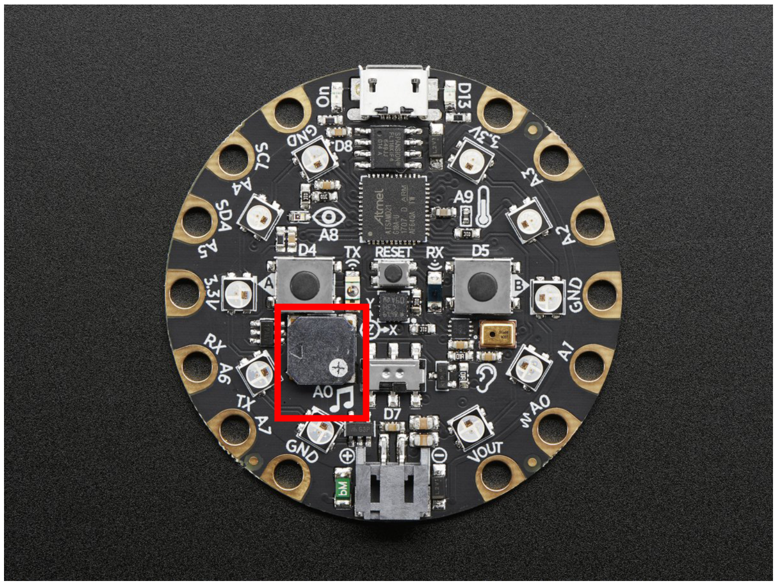
Decreasing `shake_threshold` increases shake sensitivity, i.e. the code will return a shake detected more easily with a lower `shake_threshold`. Increasing it causes the opposite. `shake_threshold` requires a minimum value of 10 - 10 is the value when the board is not moving, therefore anything less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



To use with the Circuit Playground Express:

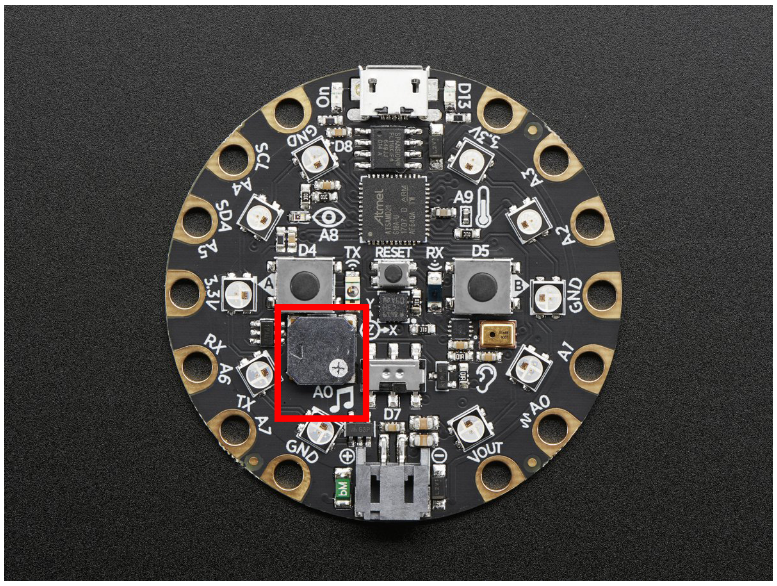
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

stop_tone()

Use with start_tone to stop the tone produced.



To use with the Circuit Playground Express:

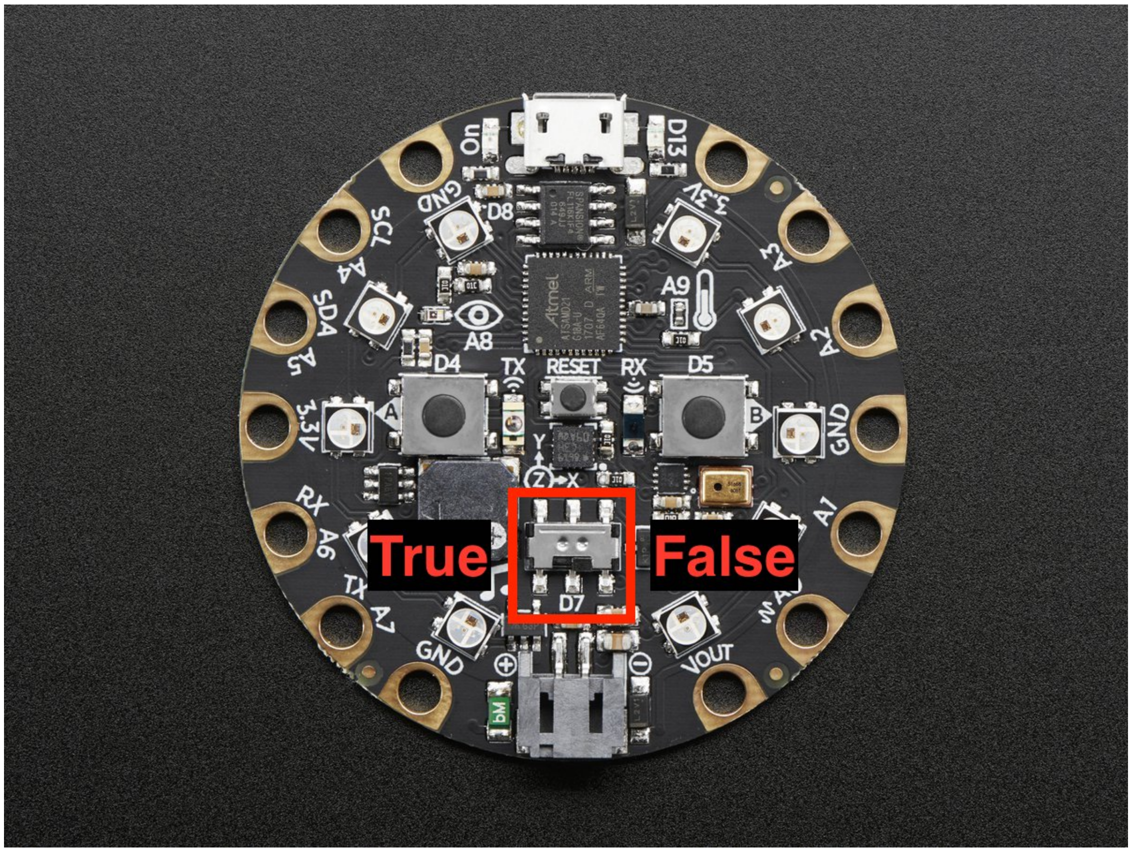
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.button_a:
        cpb.start_tone(262)
    elif cpb.button_b:
        cpb.start_tone(294)
    else:
        cpb.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.



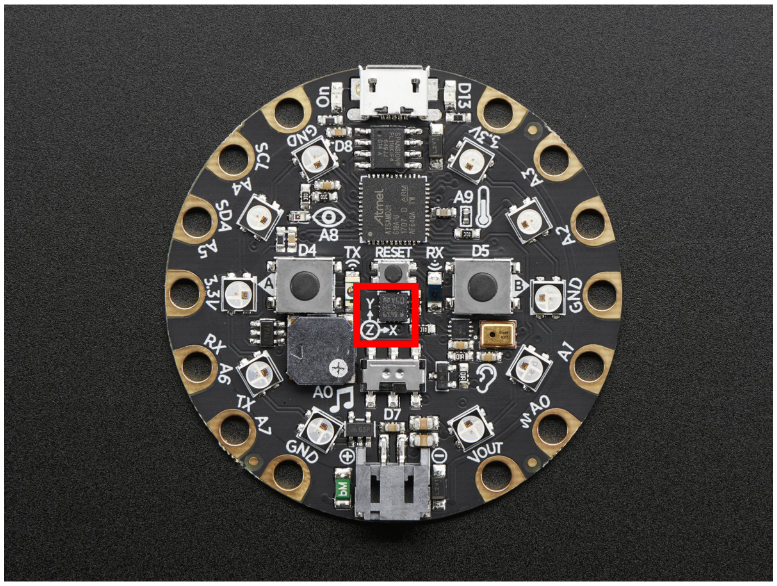
To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx
import time

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb
import time

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

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



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

To use with Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

cpx.detect_taps = 1

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

To use with Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

cpb.detect_taps = 1

while True:
    if cpb.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!")
```

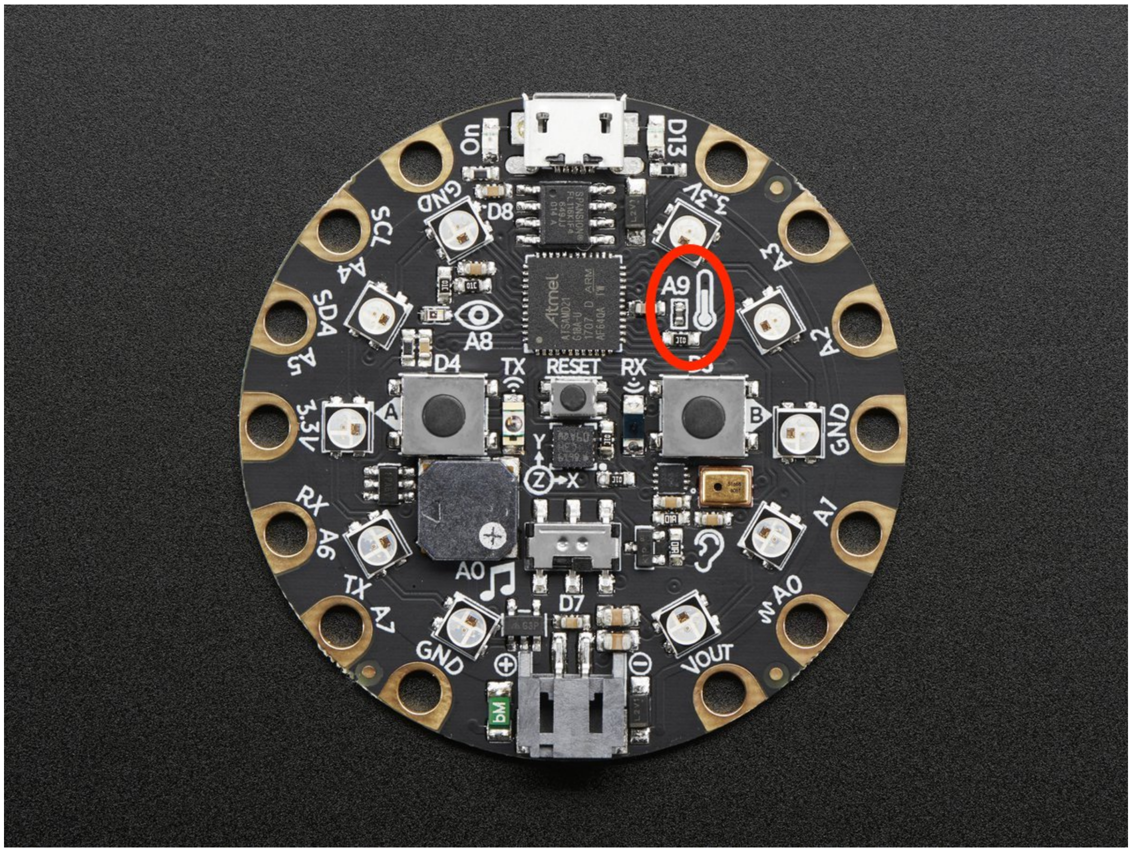
(continues on next page)

(continued from previous page)

```
# 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 in Celsius.



Converting this to Fahrenheit is easy!

To use with the Circuit Playground Express:

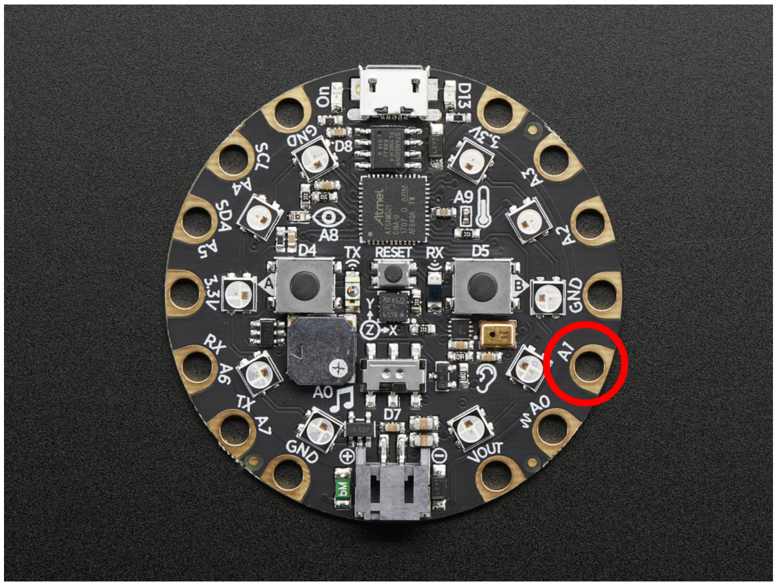
```
from adafruit_circuitplayground.express import cpx
import time

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)
```

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb
import time

while True:
    temperature_c = cpb.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.



To use with the Circuit Playground Express:

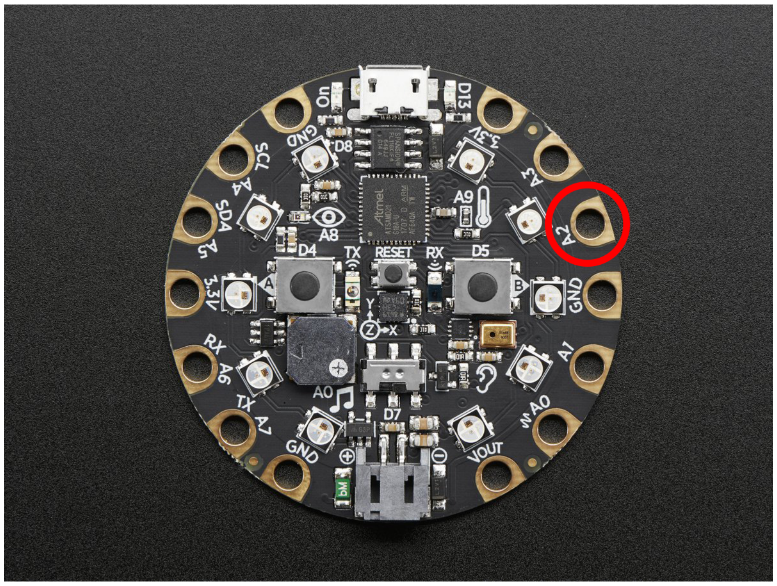
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_A2

Detect touch on capacitive touch pad A2.



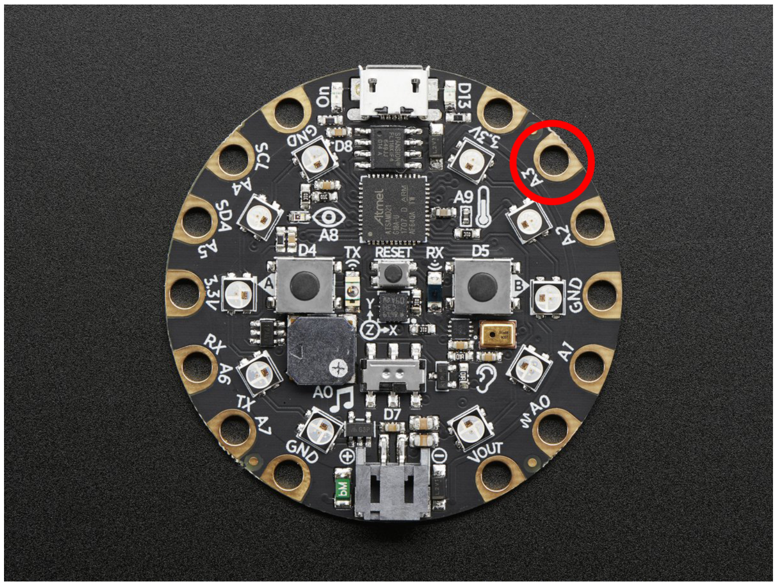
To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_A3
Detect touch on capacitive touch pad A3.



To use with the Circuit Playground Express:

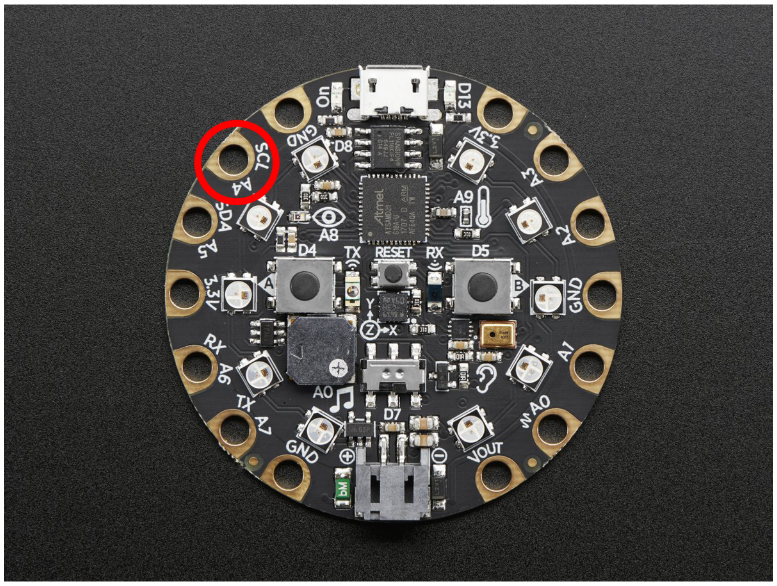
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_A4

Detect touch on capacitive touch pad A4.



To use with the Circuit Playground Express:

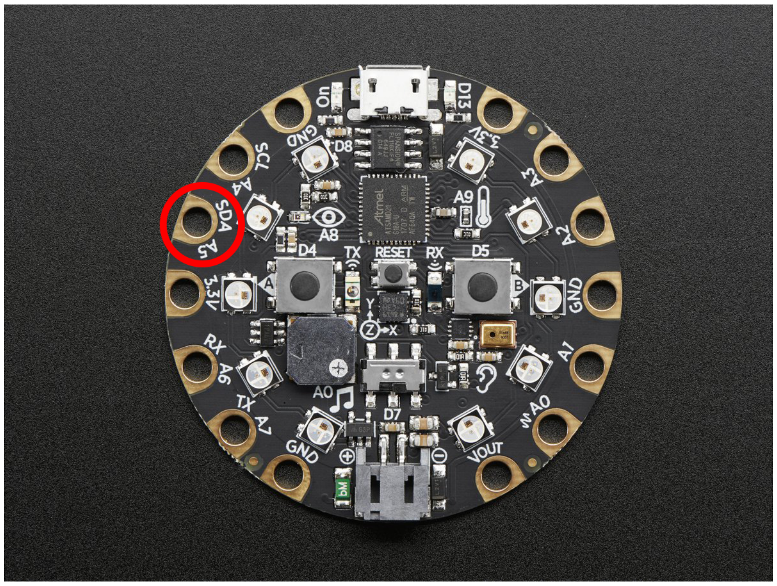
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_A5

Detect touch on capacitive touch pad A5.



To use with the Circuit Playground Express:

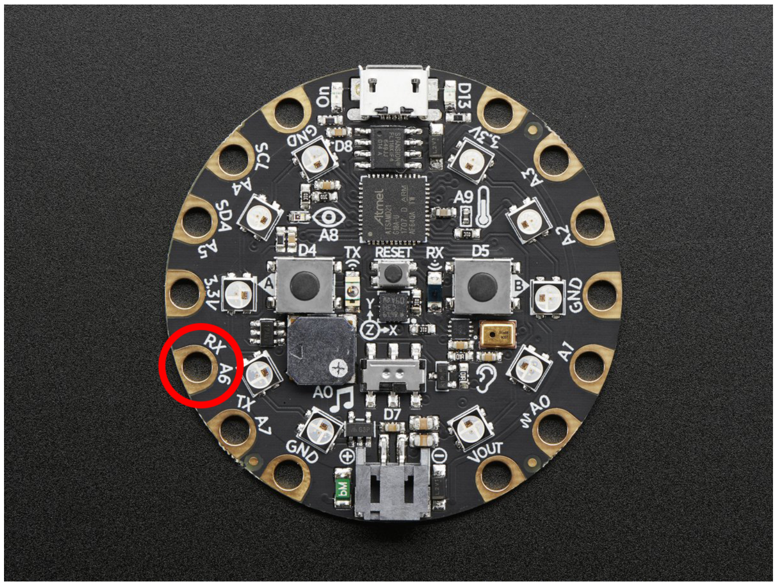
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_A6

Detect touch on capacitive touch pad A6.



To use with the Circuit Playground Express:

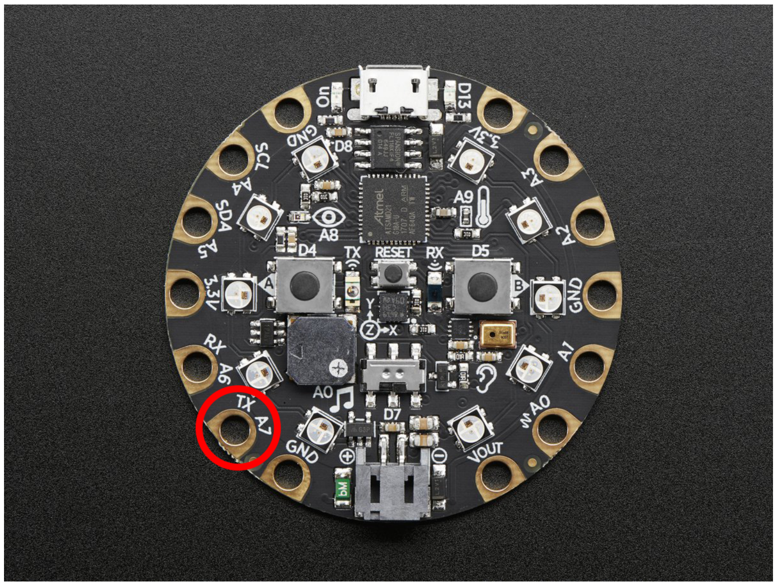
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

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

touch_TX

Detect touch on capacitive touch pad TX (also known as A7 on the Circuit Playground Express) Note: can be called as `touch_A7` on Circuit Playground Express.



To use with the Circuit Playground Express:

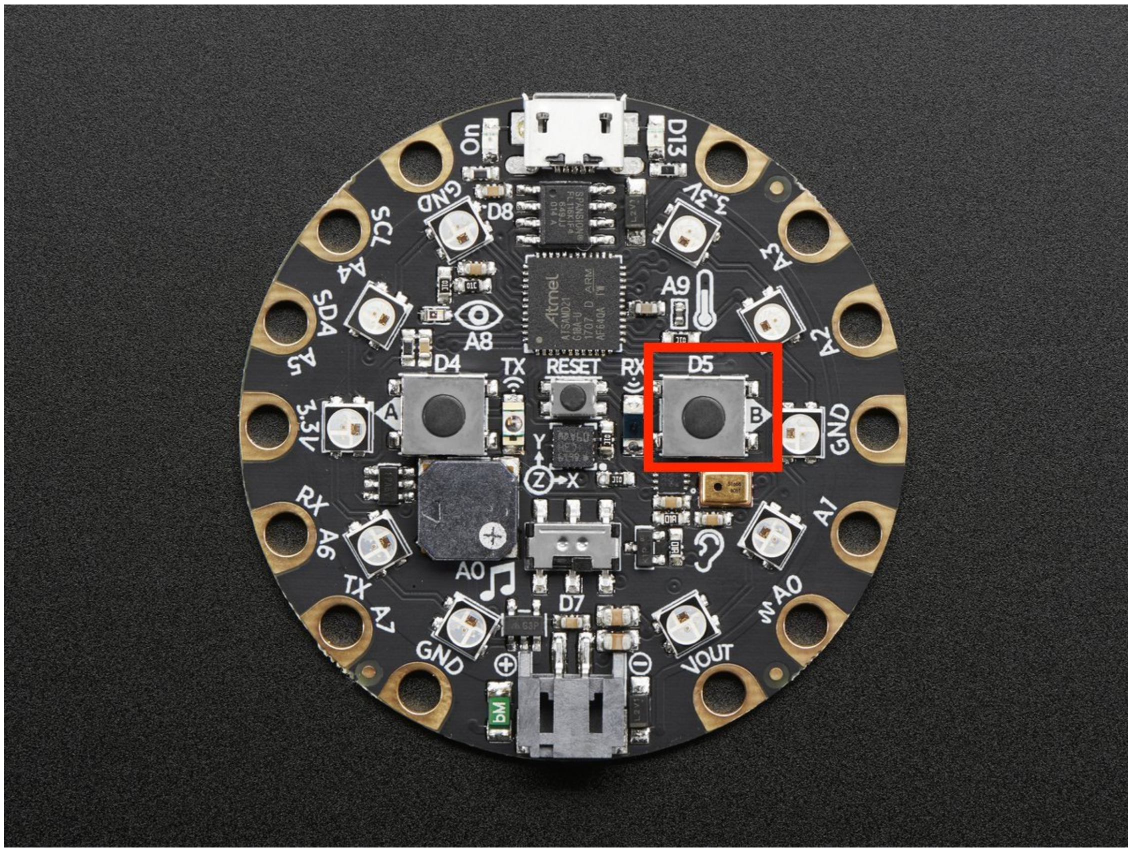
```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.touch_TX:
        print('Touched pad TX')
```

were_pressed

Returns a set of the buttons that have been pressed



To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

while True:
    print(cpx.were_pressed)
```

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    print(cpb.were_pressed)
```

class adafruit_circuitplayground.circuit_playground_base.**Photocell**(*pin*)
Simple driver for analog photocell on the Circuit Playground Express and Bluefruit.

light
Light level.

5.3 adafruit_circuitplayground.bluefruit

CircuitPython helper for Circuit Playground Bluefruit.

- Author(s): Kattni Rembor

5.3.1 Implementation Notes

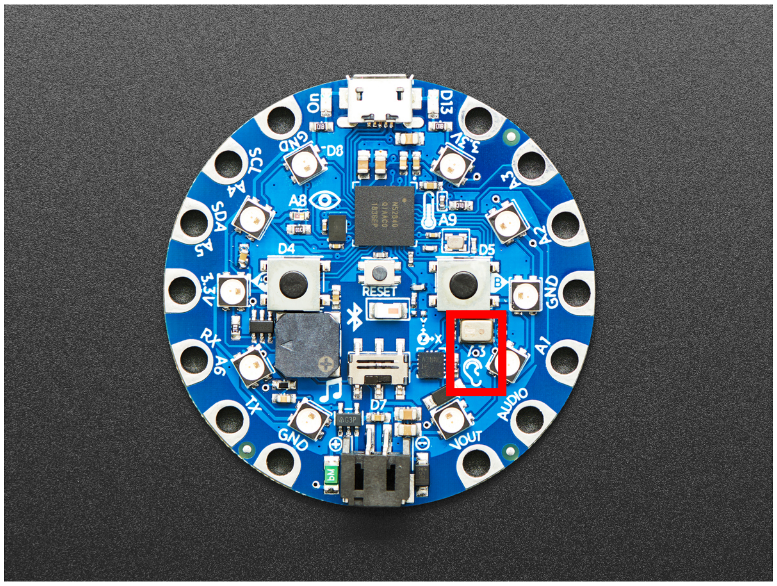
Hardware:

- Circuit Playground Bluefruit

class adafruit_circuitplayground.bluefruit.**Bluefruit**
Represents a single CircuitPlayground Bluefruit.

loud_sound(*sound_threshold=200*)
Utilise a loud sound as an input.

Parameters **sound_threshold** (*int*) – Threshold sound level must exceed to return true (Default: 200)



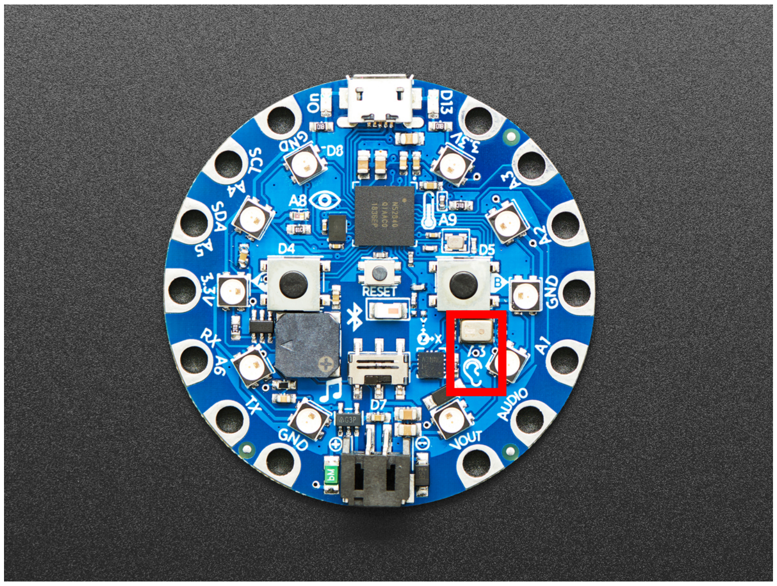
This example turns the LEDs red each time you make a loud sound. Try clapping or blowing onto the microphone to trigger it.

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.loud_sound():
        cpb.pixels.fill((50, 0, 0))
    else:
        cpb.pixels.fill(0)
```

You may find that the code is not responding how you would like. If this is the case, you can change the loud sound threshold to make it more or less responsive. Setting it to a higher number means it will take a louder sound to trigger. Setting it to a lower number will take a quieter sound to trigger. The following example shows the threshold being set to a higher number than the default.

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.loud_sound(sound_threshold=300):
        cpb.pixels.fill((50, 0, 0))
    else:
        cpb.pixels.fill(0)
```

`sound_level`
Obtain the sound level from the microphone (sound sensor).



This example prints the sound levels. Try clapping or blowing on the microphone to see the levels change.

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    print(cpb.sound_level)
```

`adafruit_circuitplayground.bluefruit.`**`cpb = <adafruit_circuitplayground.bluefruit.Bluefruit`**
Object that is automatically created on import.

To use, simply import it from the module:

```
from adafruit_circuitplayground.bluefruit import cpb
```

5.4 `adafruit_circuitplayground.express`

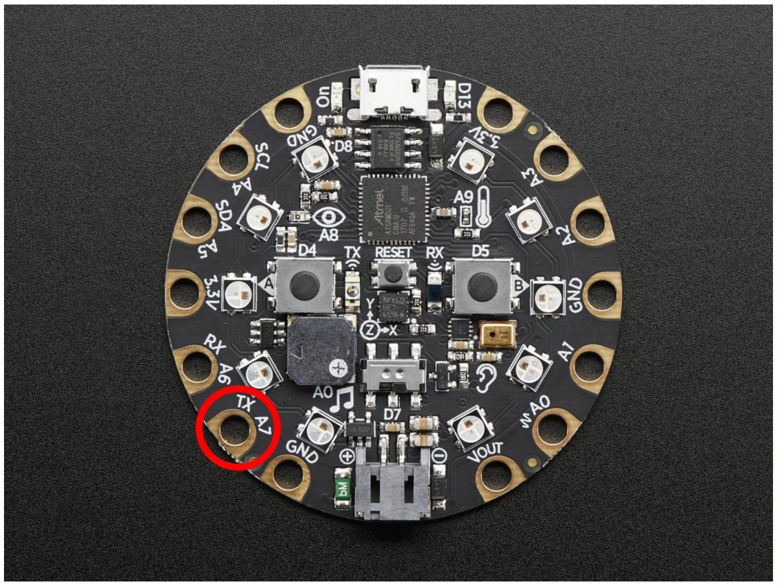
CircuitPython helper for Circuit Playground Express.

Hardware:

- Circuit Playground 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.

`touch_A7`
Detect touch on capacitive touch pad TX (also known as A7 on the Circuit Playground Express) Note: can be called as `touch_A7` on Circuit Playground Express.



To use with the Circuit Playground Express:

```
from adafruit_circuitplayground.express import cpx

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

To use with the Circuit Playground Bluefruit:

```
from adafruit_circuitplayground.bluefruit import cpb

while True:
    if cpb.touch_TX:
        print('Touched pad TX')
```

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

Python Module Index

a

adafruit_circuitplayground.bluefruit, 46
adafruit_circuitplayground.circuit_playground_base, 25
adafruit_circuitplayground.express, 48

Index

A

acceleration (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 25

adafruit_circuitplayground.bluefruit (*module*), 46

adafruit_circuitplayground.circuit_playground_base (*module*), 25

adafruit_circuitplayground.express (*module*), 48

adjust_touch_threshold() (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase method*), 25

B

Bluefruit (*class in adafruit_circuitplayground.bluefruit*), 47

button_a (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 26

button_b (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 27

C

CircuitPlaygroundBase (*class in adafruit_circuitplayground.circuit_playground_base*), 25

cpb (*in module adafruit_circuitplayground.bluefruit*), 48

cpx (*in module adafruit_circuitplayground.express*), 49

D

detect_taps (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 28

E

Express (*class in adafruit_circuitplayground.express*), 48

L

light (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 29

light (*adafruit_circuitplayground.circuit_playground_base.Photocell attribute*), 46

loud_sound() (*adafruit_circuitplayground.bluefruit.Bluefruit method*), 47

P

Photocell (*class in adafruit_circuitplayground.circuit_playground_base*), 46

pixels (*adafruit_circuitplayground.circuit_playground_base.CircuitPlay attribute*), 30

play_file() (*adafruit_circuitplayground.circuit_playground_base.Circ method*), 31

play_tone() (*adafruit_circuitplayground.circuit_playground_base.Circ method*), 32

R

red_led (*adafruit_circuitplayground.circuit_playground_base.CircuitPla attribute*), 33

S

shake() (*adafruit_circuitplayground.circuit_playground_base.CircuitPla method*), 34

sound_level (*adafruit_circuitplayground.bluefruit.Bluefruit attribute*), 47

start_tone() (*adafruit_circuitplayground.circuit_playground_base.Ci method*), 35

stop_tone() (*adafruit_circuitplayground.circuit_playground_base.Circ method*), 36

switch (*adafruit_circuitplayground.circuit_playground_base.CircuitPlay attribute*), 37

T

tapped (*adafruit_circuitplayground.circuit_playground_base.CircuitPlay attribute*), 38

temperature (*adafruit_circuitplayground.circuit_playground_base.Circ attribute*), 40

touch_A1 (*adafruit_circuitplayground.circuit_playground_base.CircuitPl attribute*), 41

touch_A2 (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 41
touch_A3 (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 42
touch_A4 (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 43
touch_A5 (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 43
touch_A6 (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 44
touch_A7 (*adafruit_circuitplayground.express.Express attribute*), 48
touch_TX (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 45

W

were_pressed (*adafruit_circuitplayground.circuit_playground_base.CircuitPlaygroundBase attribute*), 45