

Programmable Instrumentation with Circuit Python

Head Neuroengineer:
Ed Soucy PhD



Neuroengineer:
Brett Graham PhD



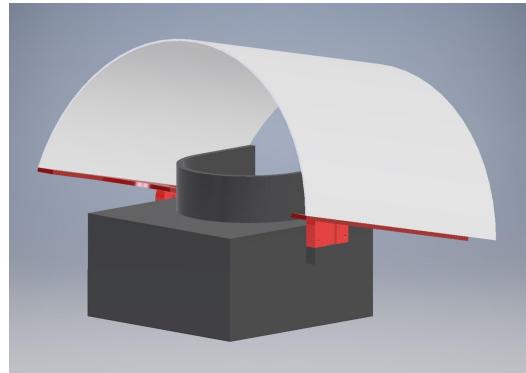
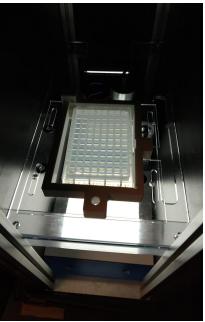
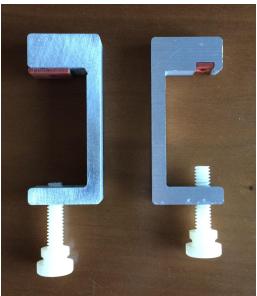
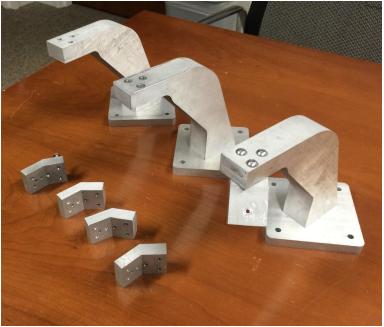
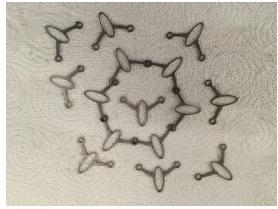
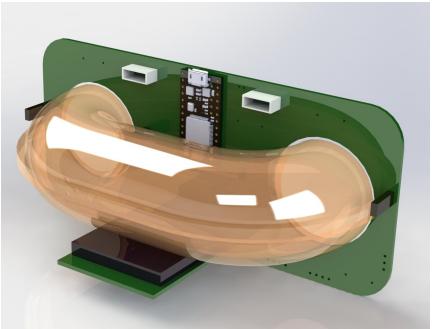
Harvard Center for Brain Science Neurotechnology Core

Technology can help advance science

The Neurotechnology core offers:

- A modern fabrication and prototyping shop (makerspace)
- Training on tools, instrument design, data analysis and general scientific consulting
- Design and fabrication of novel instrumentation

We are generalist, scientist, engineers that help will all technical aspects of science



Example: Measuring reaction time

How quickly can a person press a button when queued by a light flash or tone?

How might we measure this?

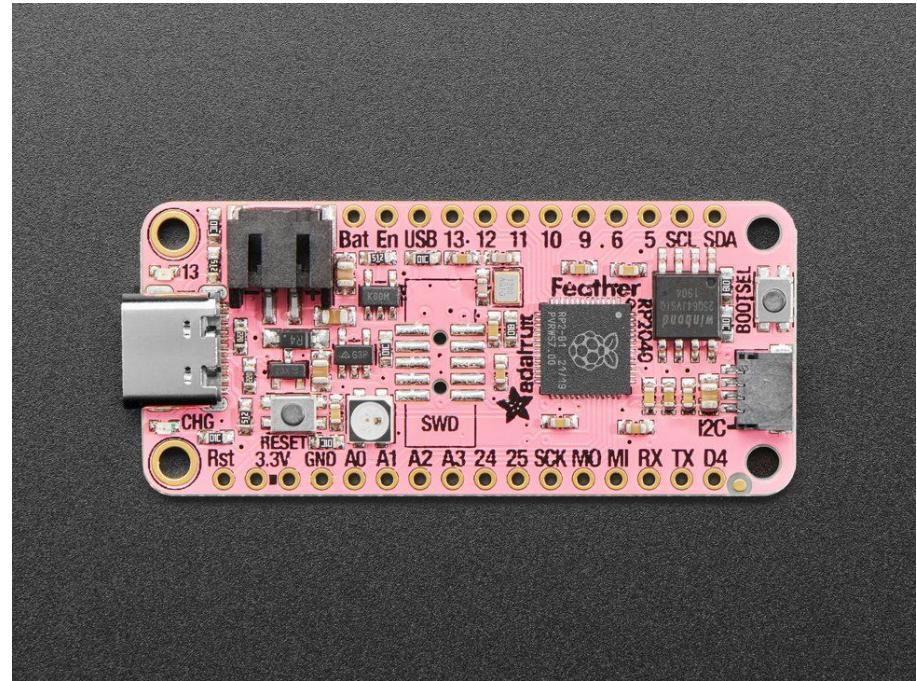
What is a microcontroller? (RP2040)

Small **programmable** computer

- Dual core ~125 MHz
- 264 kB RAM
- 8 MB storage (Flash)

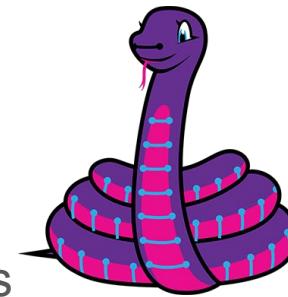
No OS = more control

Lots of ‘low level’ IO



What is CircuitPython?

Adaptation of Python language to microcontrollers



Officially supported by Adafruit electronics (adafruit.com)

<https://docs.circuitpython.org/en/7.3.x/README.html>

Alternative to Arduino (which uses C/C++) boards can often be used with either

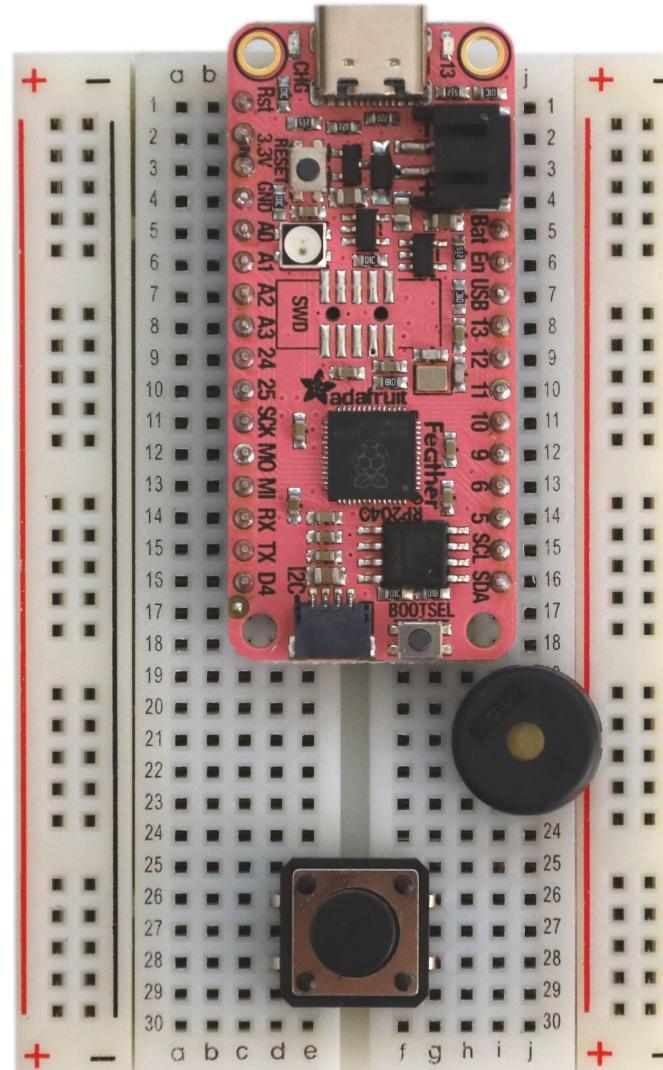
Any text editor works, we'll use Mu: <https://codewith.mu/>



Building your instrument

Solderless Breadboard

Good for testing your circuit

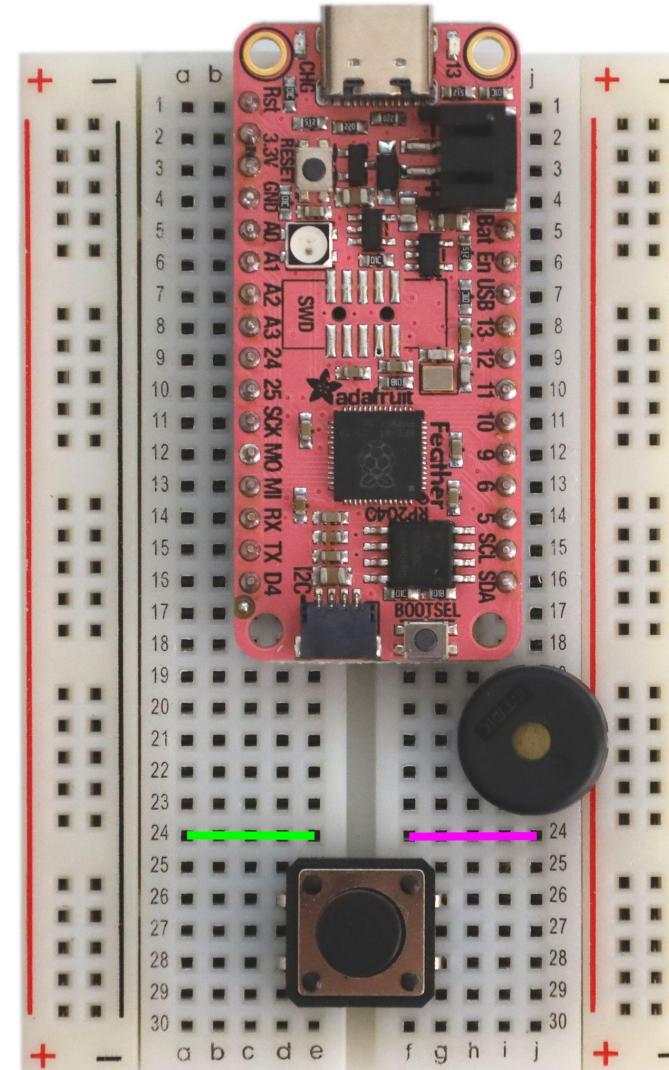


Breadboard connections

Rows of sockets are connected

24 a b c d e : all connected (green)

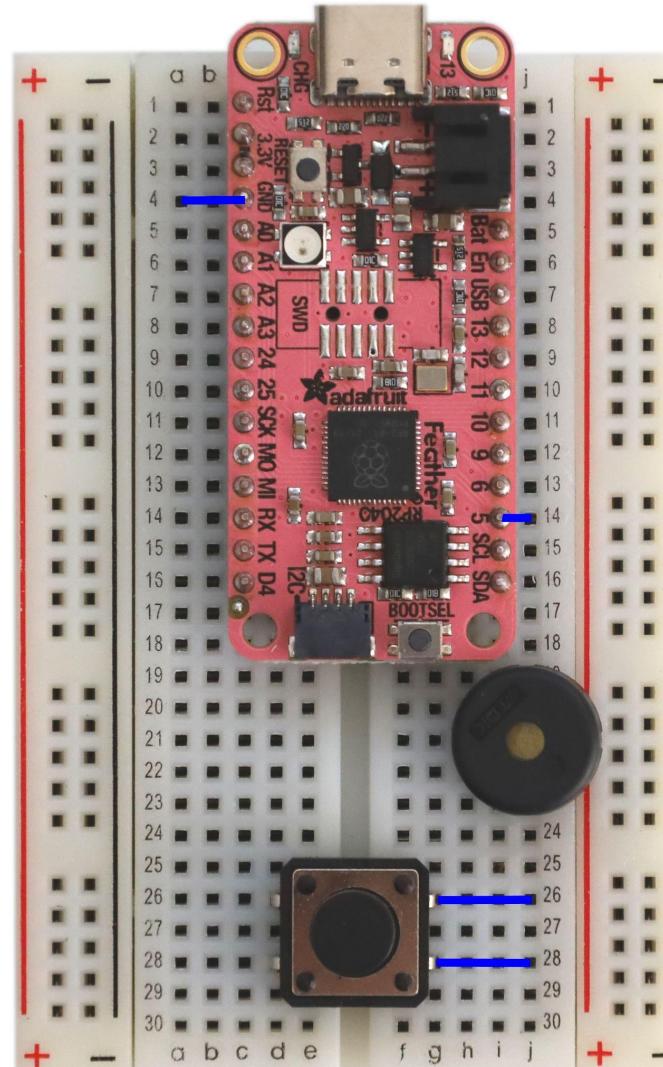
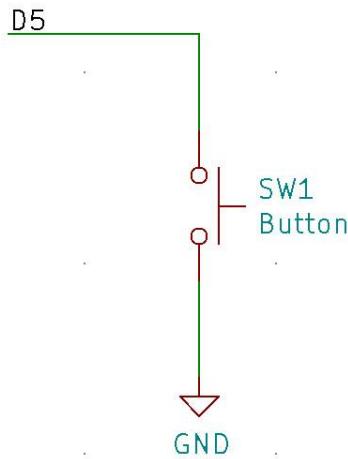
24 f g h i j : all connected (pink)



Button circuit

Requires 2 connections:

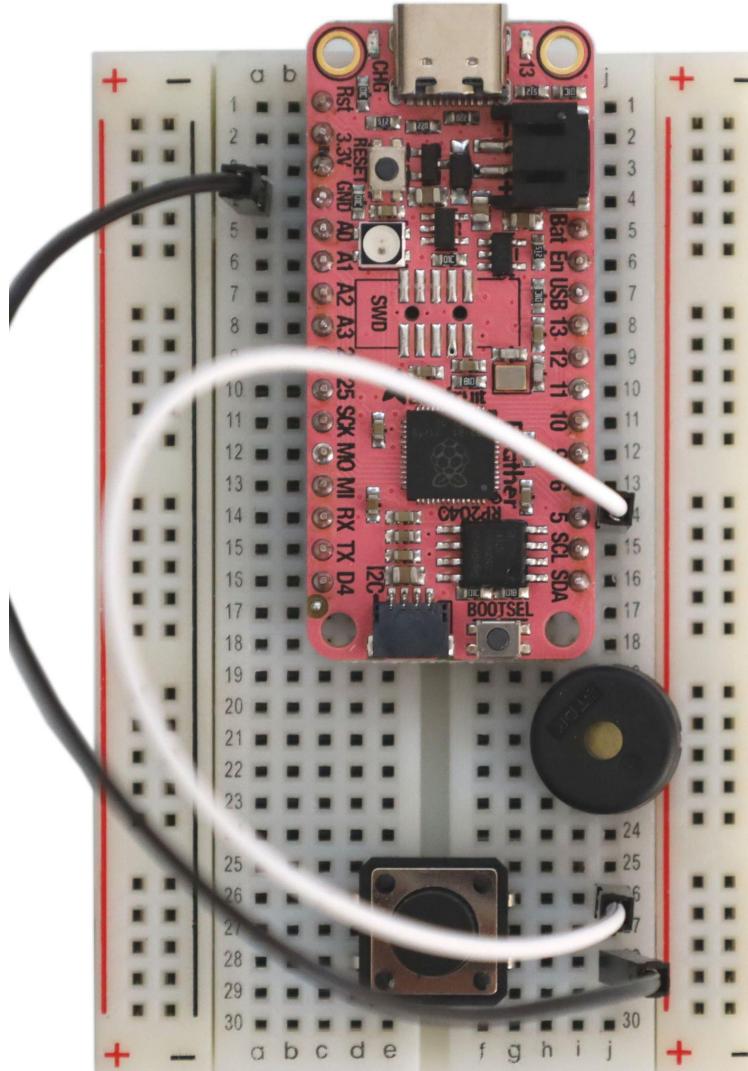
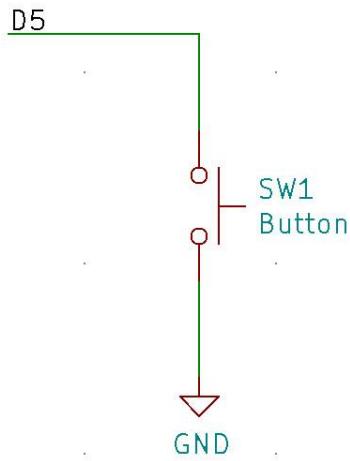
- Digital input (pin 5)
- Ground (GND)



Wiring

Connect 2 wires

- 4 a to 28 j
- 14 j to 26 j



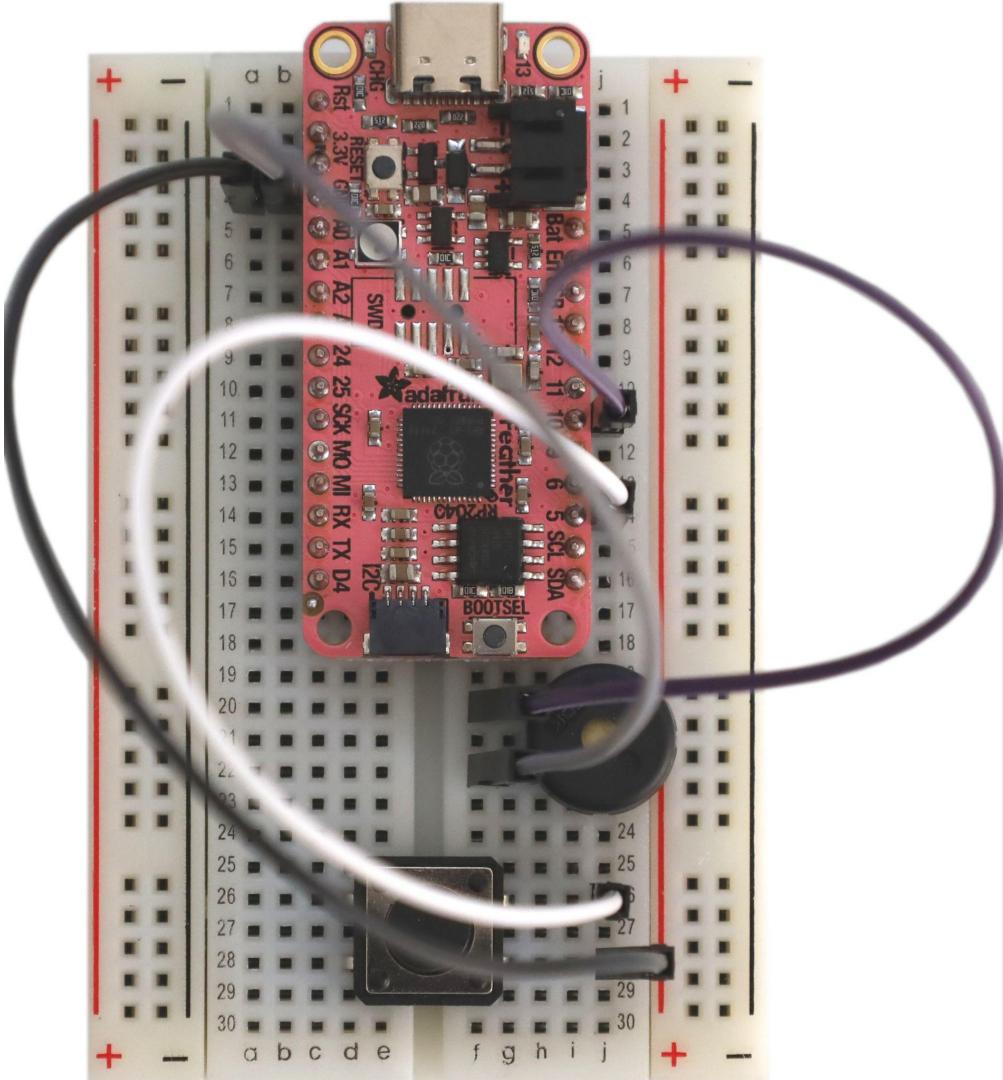
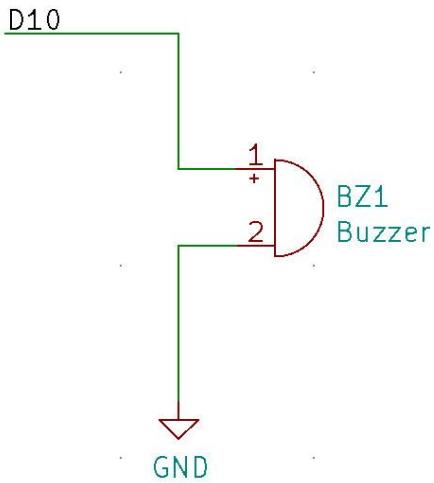
Buzzer circuit

Requires 2 connections

- Digital output (pin 10)
- Ground (GND)

Run 2 wires

- 11 j to 20 f
- 4 b to 22 f



Verifying your wiring

Your boards are pre-programmed to light an LED when the button is pressed

Wire up your board, plug it into your computer and give it a try!

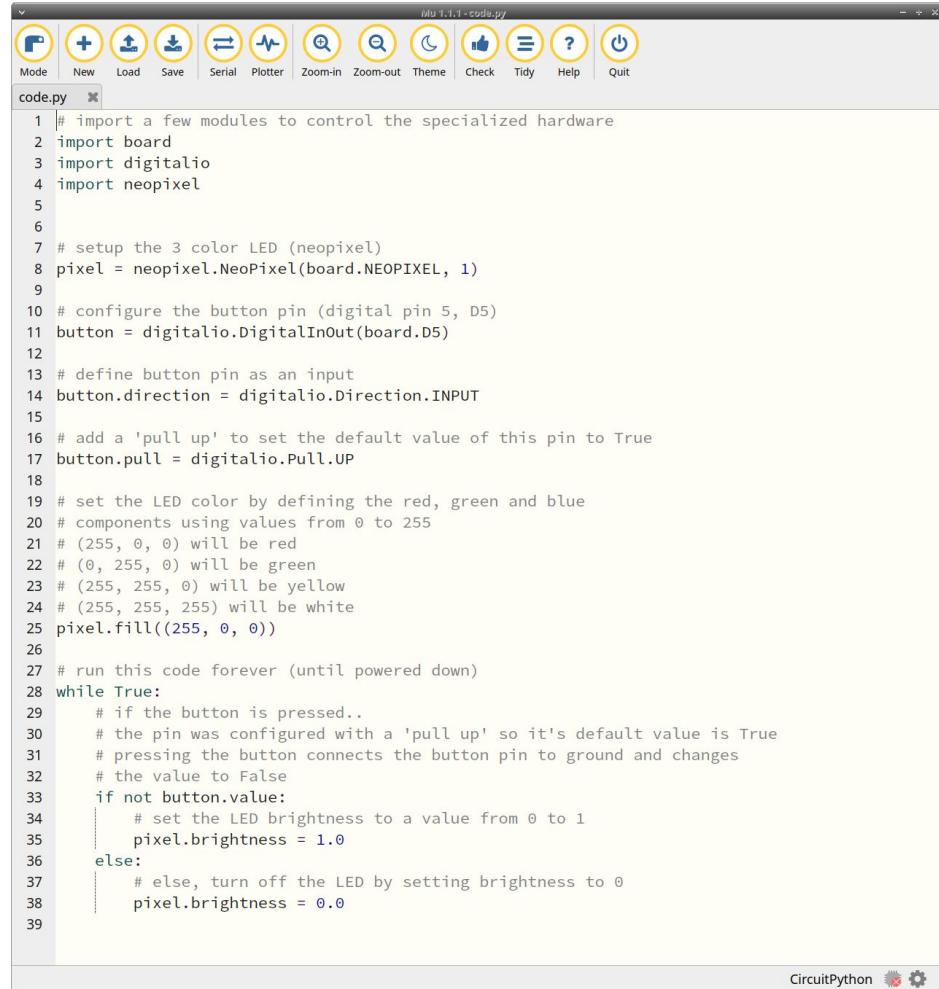
Programming

Appears as USB drive

'code.py' is plain text

Editing this text automatically resets the device (if all goes well)

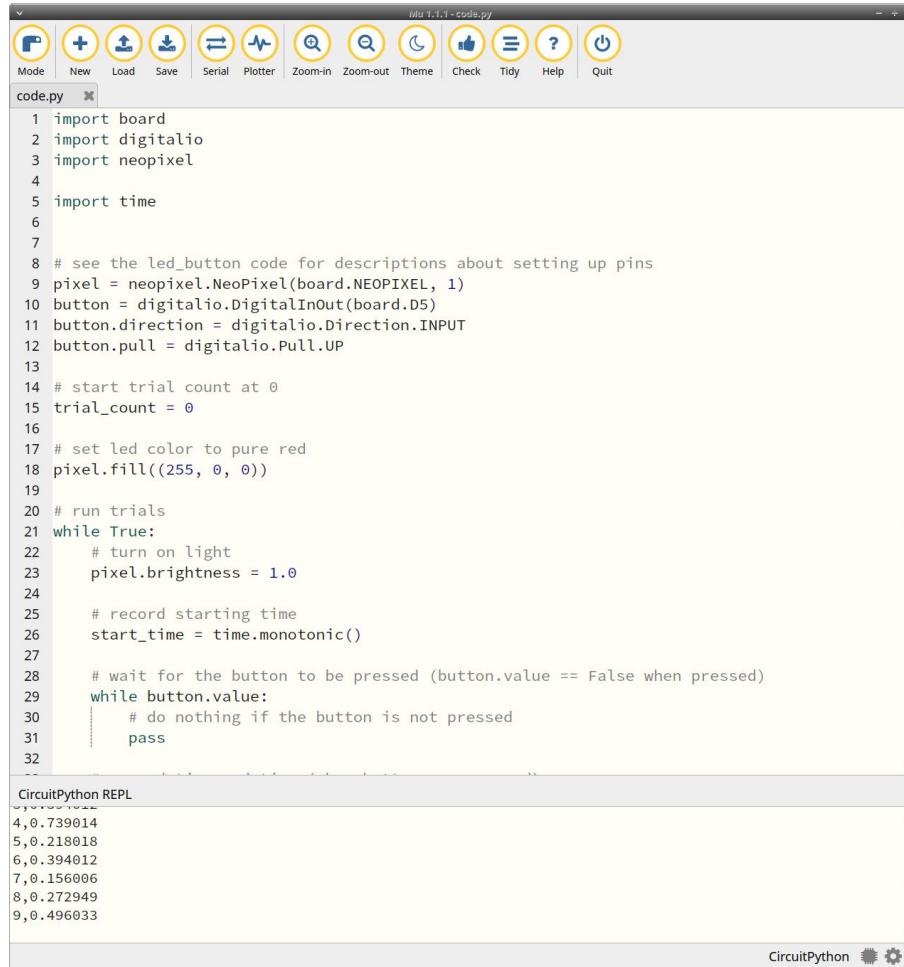
Also appears as a 'serial' device which will report errors and other text output (like your measured reaction times)



The screenshot shows the Mu code editor interface. The title bar reads "Mu 1.1.1 - code.py". The menu bar includes "Mode", "New", "Load", "Save", "Serial", "Plotter", "Zoom-in", "Zoom-out", "Theme", "Check", "Tidy", "Help", and "Quit". The main window displays the following Python code:

```
1 # import a few modules to control the specialized hardware
2 import board
3 import digitalio
4 import neopixel
5
6
7 # setup the 3 color LED (neopixel)
8 pixel = neopixel.NeoPixel(board.NEOPIXEL, 1)
9
10 # configure the button pin (digital pin 5, D5)
11 button = digitalio.DigitalInOut(board.D5)
12
13 # define button pin as an input
14 button.direction = digitalio.Direction.INPUT
15
16 # add a 'pull up' to set the default value of this pin to True
17 button.pull = digitalio.Pull.UP
18
19 # set the LED color by defining the red, green and blue
20 # components using values from 0 to 255
21 # (255, 0, 0) will be red
22 # (0, 255, 0) will be green
23 # (255, 255, 0) will be yellow
24 # (255, 255, 255) will be white
25 pixel.fill((255, 0, 0))
26
27 # run this code forever (until powered down)
28 while True:
29     # if the button is pressed..
30     # the pin was configured with a 'pull up' so it's default value is True
31     # pressing the button connects the button pin to ground and changes
32     # the value to False
33     if not button.value:
34         # set the LED brightness to a value from 0 to 1
35         pixel.brightness = 1.0
36     else:
37         # else, turn off the LED by setting brightness to 0
38         pixel.brightness = 0.0
39
```

In the bottom right corner, there is a "CircuitPython" logo with a gear icon.



The image shows the IDLE 3.3.1 Python editor interface running on a Linux system. The window title is "idle 3.3.1 - code.py". The menu bar includes File, Edit, View, Insert, Format, Tools, Help, and Quit. Below the menu is a toolbar with icons for Mode, New, Load, Save, Serial, Plotter, Zoom-in, Zoom-out, Theme, Check, Tidy, Help, and Quit.

The main text area displays the following Python code:

```
1 import board
2 import digitalio
3 import neopixel
4
5 import time
6
7
8 # see the led_button code for descriptions about setting up pins
9 pixel = neopixel.NeoPixel(board.NEOPixel, 1)
10 button = digitalio.DigitalInOut(board.D5)
11 button.direction = digitalio.Direction.INPUT
12 button.pull = digitalio.Pull.UP
13
14 # start trial count at 0
15 trial_count = 0
16
17 # set led color to pure red
18 pixel.fill((255, 0, 0))
19
20 # run trials
21 while True:
22     # turn on light
23     pixel.brightness = 1.0
24
25     # record starting time
26     start_time = time.monotonic()
27
28     # wait for the button to be pressed (button.value == False when pressed)
29     while button.value:
30         # do nothing if the button is not pressed
31         pass
32
```

The bottom panel is titled "CircuitPython REPL" and contains the following output:

```
4,0.739014
5,0.218018
6,0.394012
7,0.156006
8,0.272949
9,0.496033
```

The status bar at the bottom right shows "CircuitPython" with a gear icon.

Modifications

Measure average reaction time

Does your reaction time change across trials?

Use tone instead of light

Change light color and/or brightness

Change tone frequency

What strategies might a participant use to ‘cheat’ the experiment? If you use these strategies can you change your measured reaction time?

Where to learn more about electronics?

<https://docs.circuitpython.org/en/7.3.x/README.html>

<https://learn.adafruit.com/>

<https://learn.sparkfun.com/>

<https://www.arduino.cc/>

<https://github.com/HMS-RIC/ArduinoNanocourse>

The Art of Electronics

Us:

brett_graham@fas.harvard.edu

soucy@mcb.harvard.edu