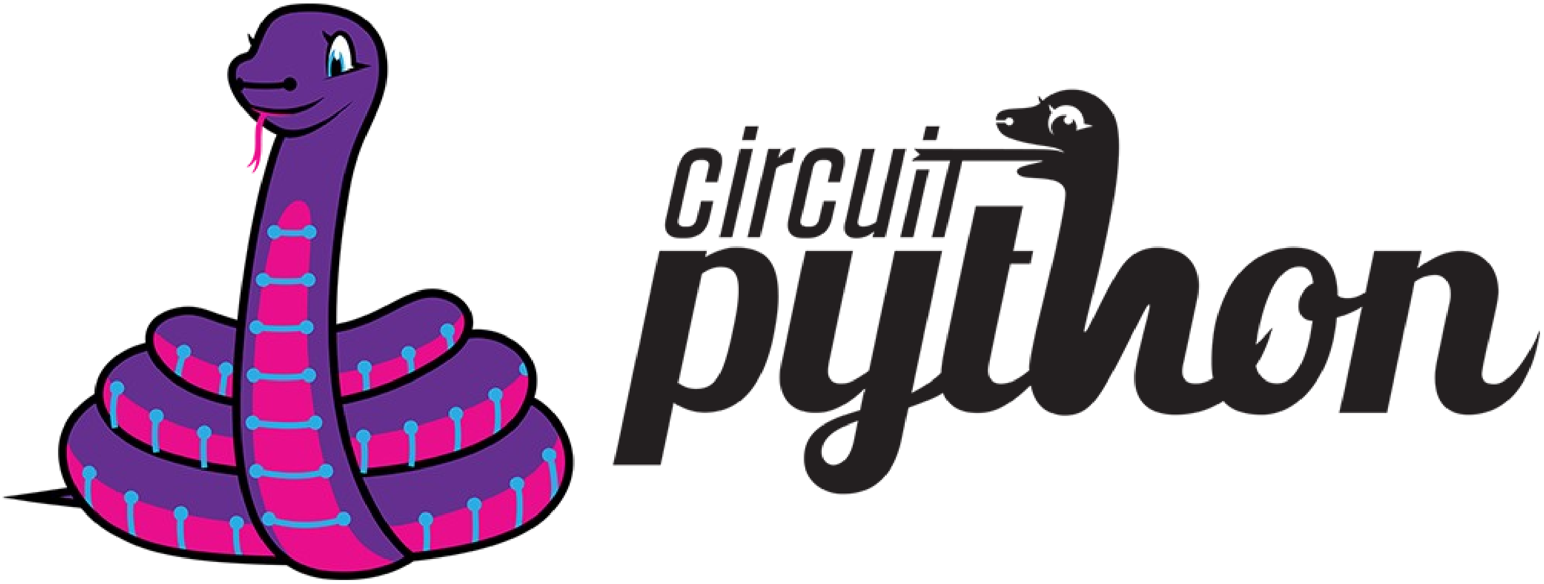
Intro to Firmware development in



- Use PyBadge as example development board w/ Rodeostat Featherwing
- Many other devevelopment boards you can use

  - over 517 on CircuitPython.org

  - feather M4 Express, QtPy, Raspberry Pi Pico, Teensy 4.1, etc.

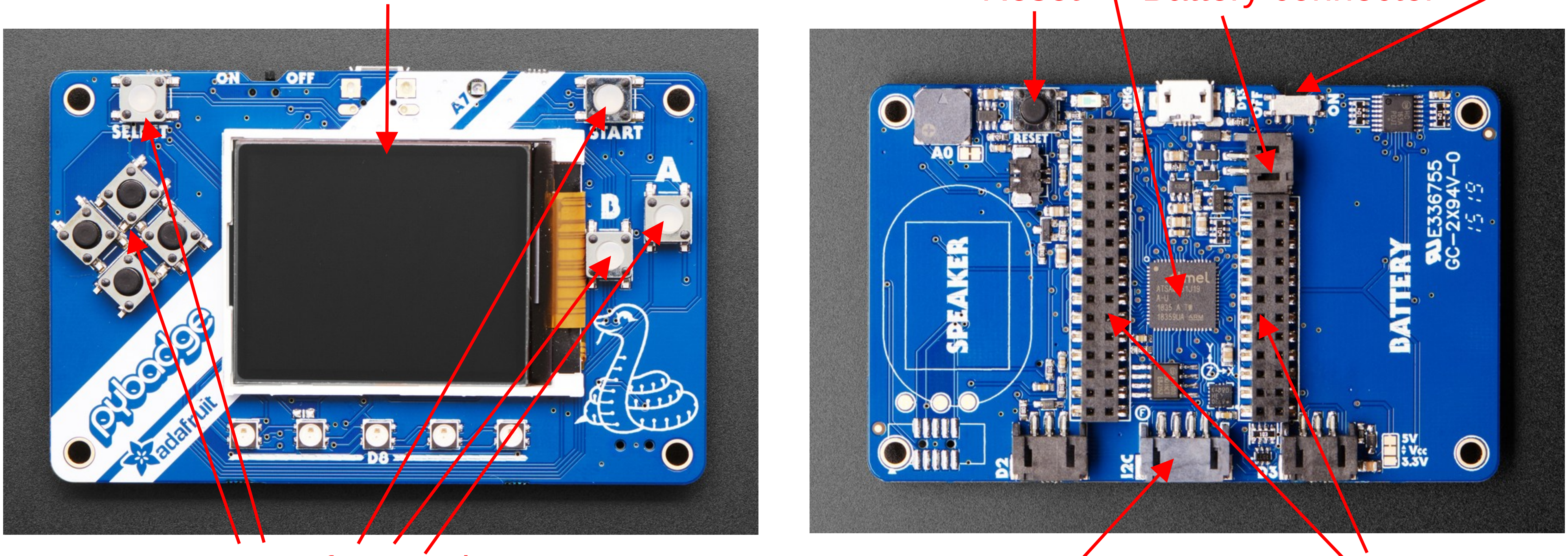
What are we going to cover?

- PyBadge hardware overview
- Setting up your board: installing bootloader & circuitpython
- Basic development tools: editor/IDE, serial terminal, how to upload firmware
- First programs: hello world, multiple source files, …
- Talking to hardware. DigitalIO, AnalogIO, etc.
- More development tools: libraries,the CircuitPython bundle and circup
- Using buttons and display
- Startup configuration: boot.py
- Serial communications and writing files
- Rodeostat Featherwing hardware and the Potentiostat class
- Potentiostat examples

Slides and examples on github

https://github.com/iorodeo/circuitpython_tutorial

PyBadge hardware overview



TFT display (160 x 128)

Buttons for user input

MCU (ATSAMD51)

Reset

Battery connector

on/off

I2C connector

Featherwing headers
DigialIO, AnalogIO,

https://learn.adafruit.com/adafruit-pybadge/overview

PyBadge hardware overview

- **MCU ATSAMD51**

  - 3.3V, 120MHz

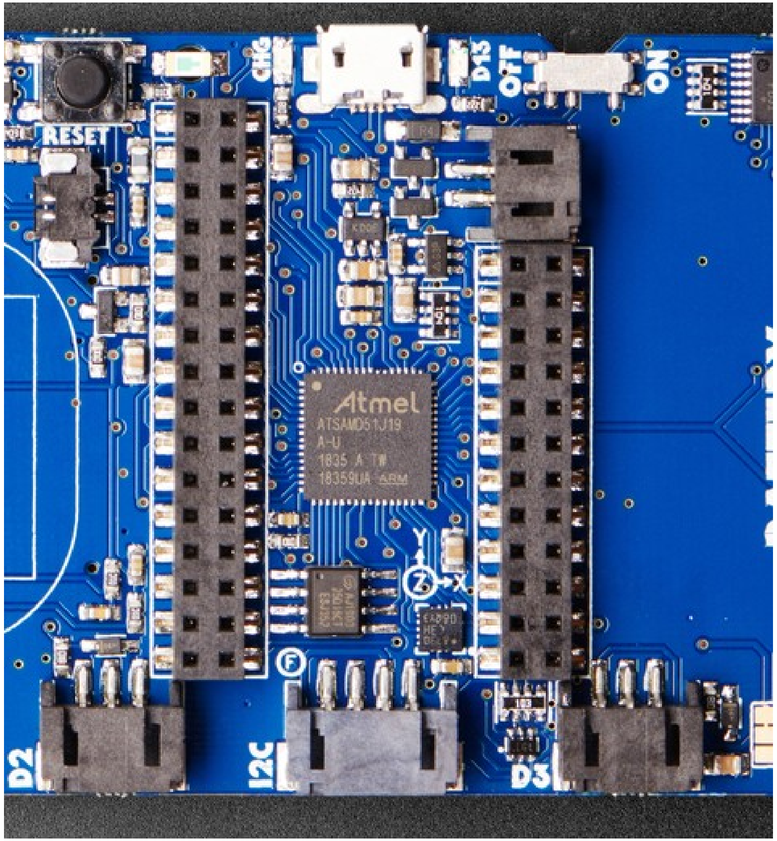
  - 192KB of RAM

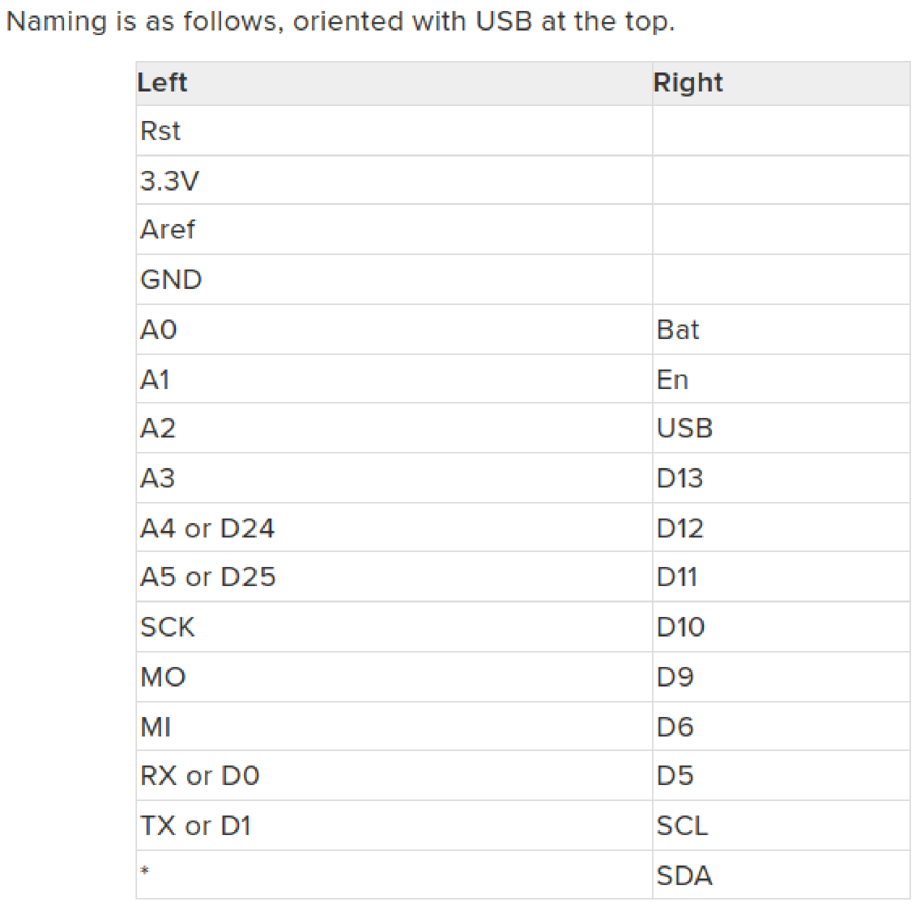
  - 512KB of flash
- **2 MB of QSPI flash for file storage**
- **13 GPIO (Digital input/outpus) D0, …, D13**
- **8 Analog Inputs (ADC) A0, … A7**
- **2 Analog Outputs (DAC) A0, A1**
- **Serial buses: I2C, SPI, UART**
- **8 user buttons**
- **TFT Display 1.8”, 160x128, color**
- **Other stuff: lipo charger, temp sensor, light sensor, accelerometer, neopixels**

https://www.adafruit.com/product/4200

https://learn.adafruit.com/adafruit-pybadge/overview

Feather header pinout




Naming is as follows, oriented with USB at the top.

Left	Right
Rst	
3.3V	
Aref	
GND	
A0	Bat
A1	En
A2	USB
A3	D13
A4 or D24	D12
A5 or D25	D11
SCK	D10
MO	D9
MI	D6
RX or D0	D5
TX or D1	SCL
*	SDA

Installing/updating CircuitPython & bootloader

- **When connected via USB PyBadge will show up as drive called CIRCUITPYTHON**

  - CircuitPython is running

- **We want the uf2 boot drive to appear, called PYBADGEBOOT**

  - unmount/safely-remove drive

  - double-click the RESET button

  - a new drive will appear named PYBADGEBOOT

- **Get new bootloader and version of CircuitPython from circuitpython.org**

  - copy bootloader PYBADGEBOOT

  - copy new circuitpython to PYBADGEBOOT

  - CIRCUITPYTHON drive should reappear

https://learn.adafruit.com/adafruit-pybadge/installing-circuitpython

Basic development tools

- **Text editor/IDE for editing source code**

  - vscode, mu editor, vim, emacs, …

- **Serial monitor for debugging/exploration**

  - vscode plugin or mu editor

  - tio, screen (linux)

  - putty (windows)

- **Upload mechanism for moving source files from PC to PyBadge**

  - upload script: bash (linux), powershell (windows)-

  - mu editor

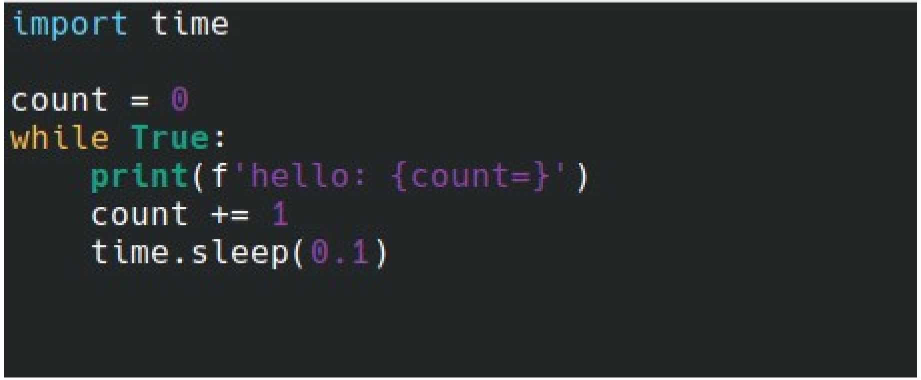
Serial monitor and REPL

- CTRL-C to stop currently running code and start REPL
- CTRL-D to reload/restart code
- help(‘modules’) to get list of built-in modules

First Program: hello world

- **entry point is code.py**

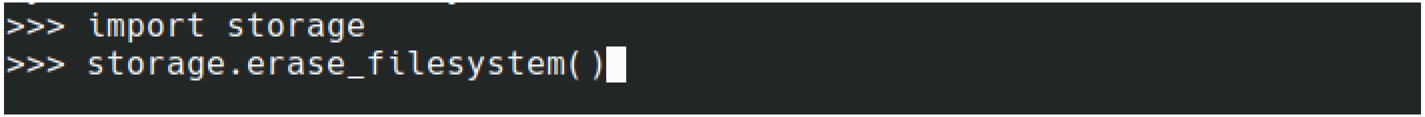
  - on startup (or reset) looks for code.py and if found executes code in file



```python
import time

count = 0
while True:
    print(f'hello: {count=}')
    count += 1
    time.sleep(0.1)
```

What if I accidentally corrupt the flash drive?

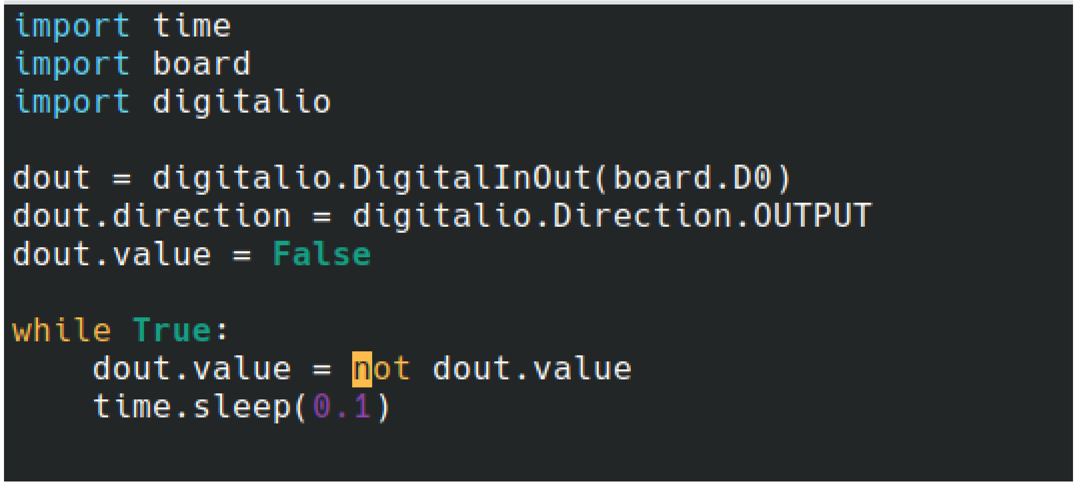
- the usual symptom is that you are unable to write to the flash drive
- can still access REPL through serial monitor
- import storage and run erase_filesystem
- Will need to reinstall project files, e.g. .py files, etc.



```
>>> import storage
>>> storage.erase_filesystem()
```

Talking to hardware: digitalio

- using built-in modules: board, digitalio
- set direction and value



```python
import time
import board
import digitalio

dout = digitalio.DigitalInOut(board.D0)
dout.direction = digitalio.Direction.OUTPUT
dout.value = False

while True:
    dout.value = not dout.value
    time.sleep(0.1)
```

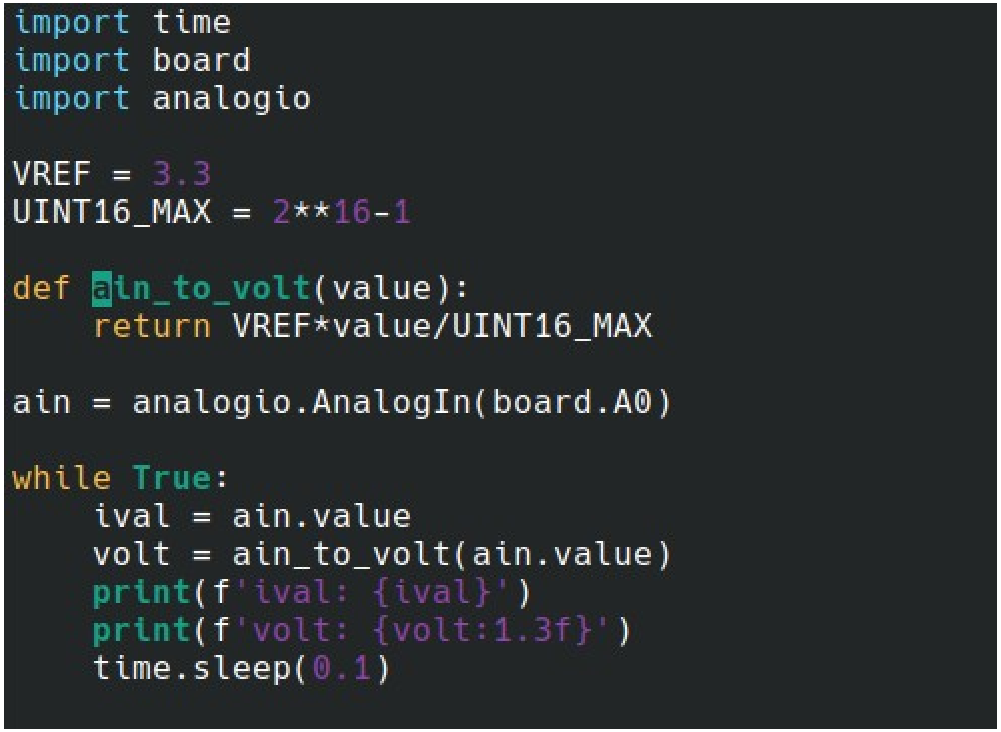
- digital input is similar

  - set Direction to INPUT

  - read .value

Talking to hardware: analogio

- using built-in modules: board, analogio
- set direction and value



```
import time
import board
import analogio

VREF = 3.3
UINT16_MAX = 2**16-1

def ain_to_volt(value):
    return VREF*value/UINT16_MAX

ain = analogio.AnalogIn(board.A0)

while True:
    ival = ain.value
    volt = ain_to_volt(ain.value)
    print(f'ival: {ival}')
    print(f'volt: {volt:1.3f}')
    time.sleep(0.1)
```

- Analog out is similar - use AnalogOut, set value instead of reading

More development tools

- the CircuitPython bundle

  - https://circuitpython.org/libraries

  - https://github.com/adafruit/Adafruit_CircuitPython_Bundle

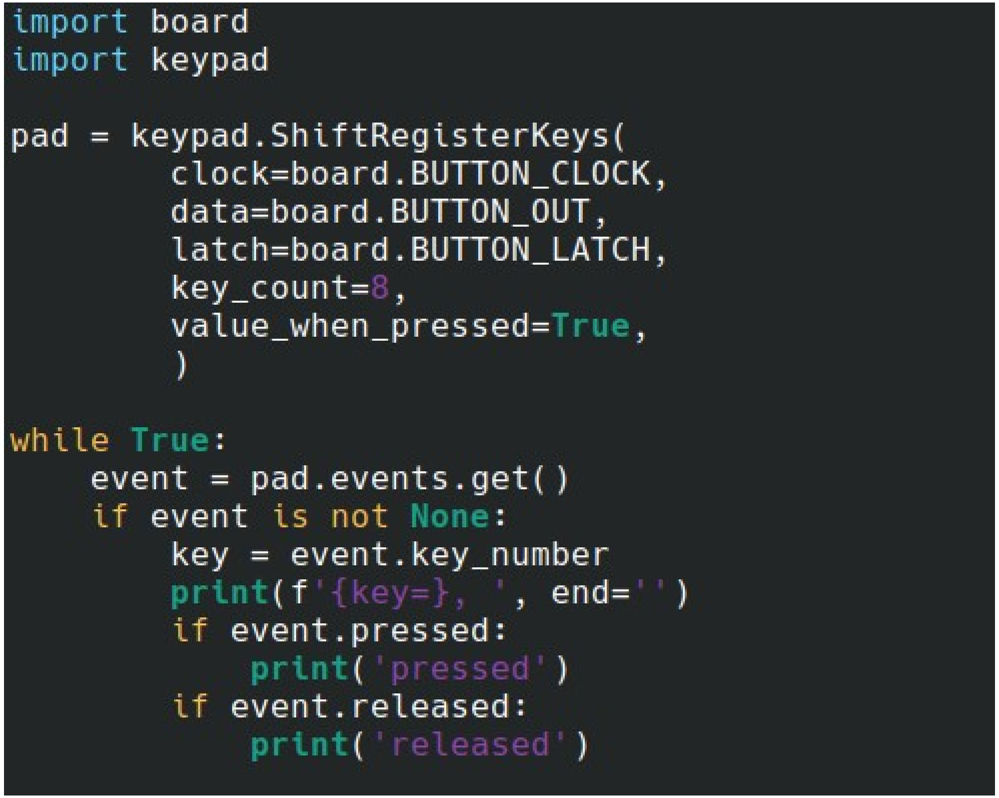
- circup: a tool for managing circuitpython libraries

  - download, update, create requirements.txt, etc.

  - https://github.com/adafruit/circup

Using buttons

- use keypad shift and look at events as they occur



```python
import board
import keypad

pad = keypad.ShiftRegisterKeys(
        clock=board.BUTTON_CLOCK,
        data=board.BUTTON_OUT,
        latch=board.BUTTON_LATCH,
        key_count=8,
        value_when_pressed=True,
        )

while True:
    event = pad.events.get()
    if event is not None:
        key = event.key_number
        print(f'{key=}, ', end='')
        if event.pressed:
            print('pressed')
        if event.released:
            print('released')
```

https://learn.adafruit.com/key-pad-matrix-scanning-in-circuitpython/shiftregisterkeys

Using the display

- **Resources**

  https://learn.adafruit.com/circuitpython-display-support-using-displayio/introduction

  https://docs.circuitpython.org/en/latest/shared-bindings/displayio/

Startup configuration

- **boot.py file is special**

  - executed when CircuitPython starts up via hard reset or powering up the board

  - not run on soft reset e.g. reload from serial console/REPL

  - can be used for configuration

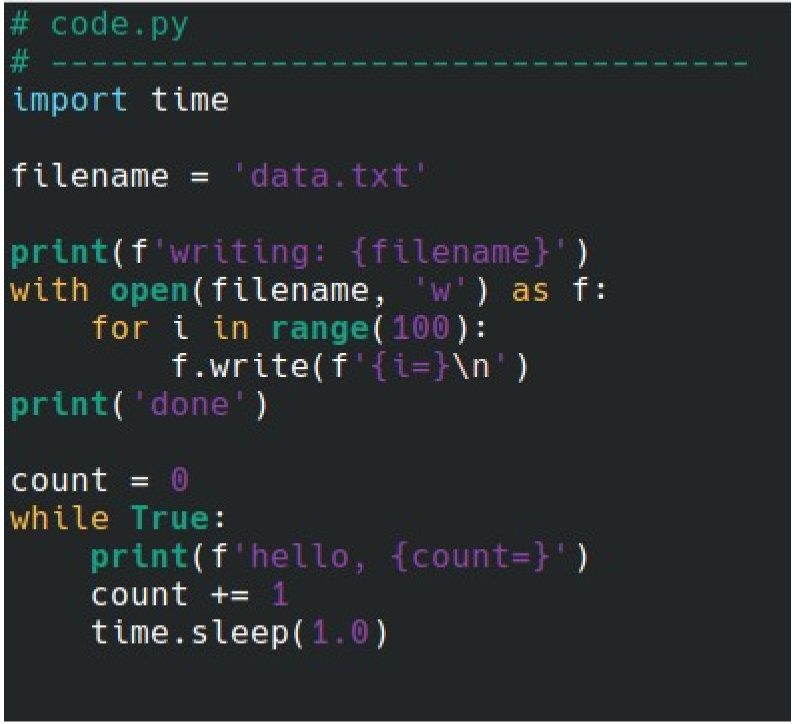
- **Example CIRCUITPY drive**

  - by default read-write by host PC and read-only for CircuitPython

  - can change so read-write for CircuitPython and read-only for host unless users presses a button on startup

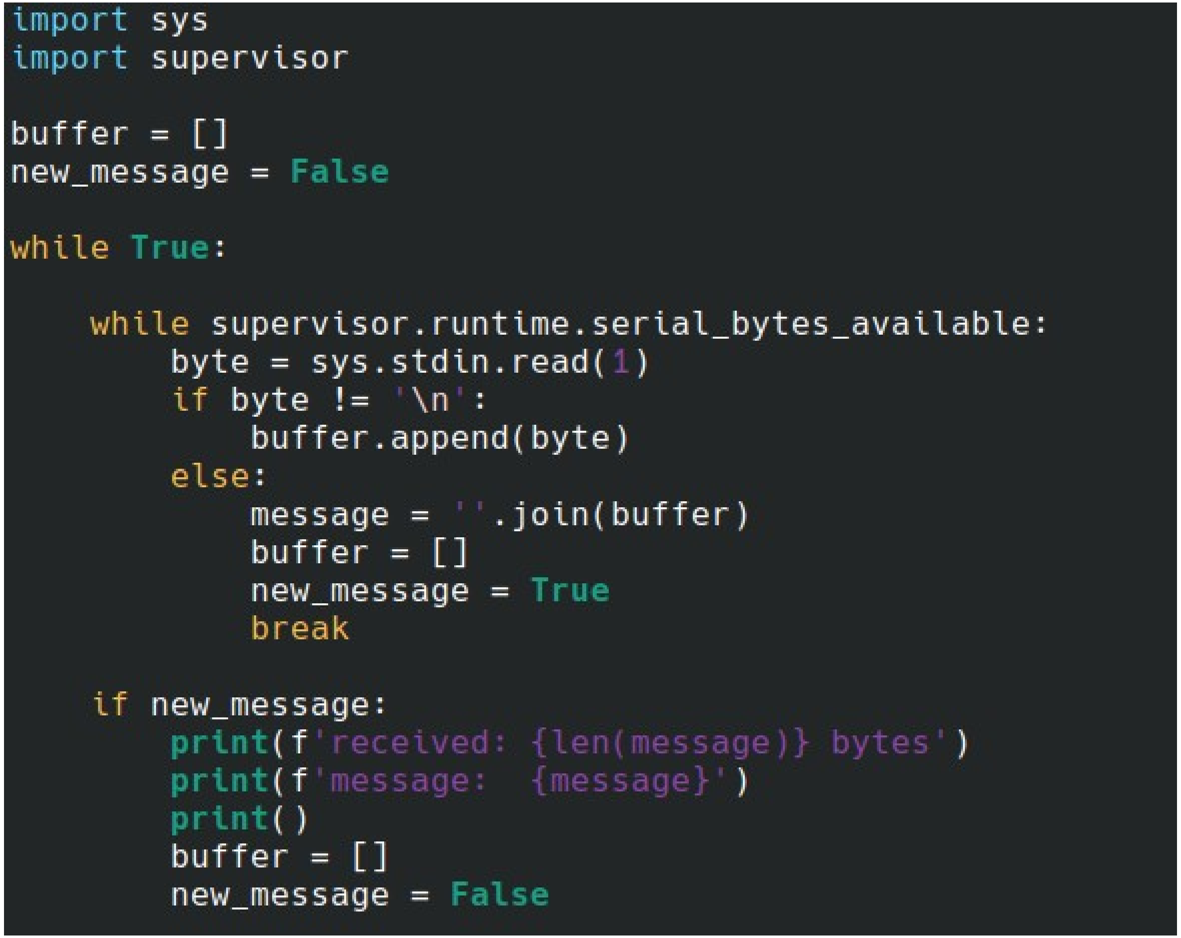
Writing files

- CIRCUITPYTON drive needs to be mounted read-write (use boot.py)



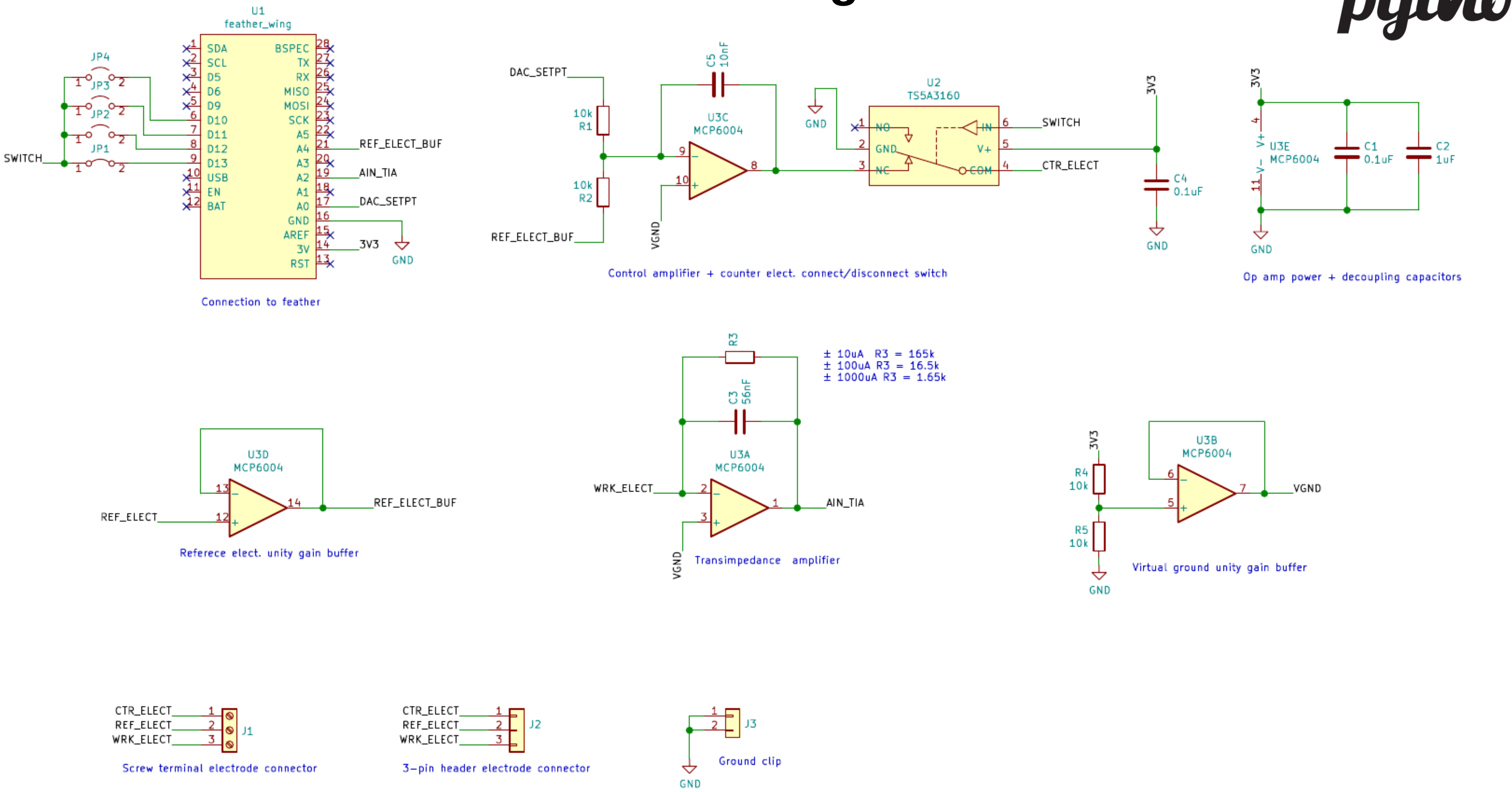
```
# code.py
# -----------------------------------
import time

filename = 'data.txt'

print(f'writing: {filename}')
with open(filename, 'w') as f:
    for i in range(100):
        f.write(f'{i=}\n')
print('done')

count = 0
while True:
    print(f'hello, {count=}')
    count += 1
    time.sleep(1.0)
```

USB/Serial communications with host



```
import sys
import supervisor

buffer = []
new_message = False

while True:

    while supervisor.runtime.serial_bytes_available:
        byte = sys.stdin.read(1)
        if byte != '\n':
            buffer.append(byte)
        else:
            message = ''.join(buffer)
            buffer = []
            new_message = True
            break

    if new_message:
        print(f'received: {len(message)} bytes')
        print(f'message:  {message}')
        print()
        buffer = []
        new_message = False
```

Rodeostat Featherwing hardware



Connection to feather

Control amplifier + counter elect. connect/disconnect switch

Op amp power + decoupling capacitors

± 10uA R3 = 165k
± 100uA R3 = 16.5k
± 1000uA R3 = 1.65k

Referece elect. unity gain buffer

Transimpedance amplifier

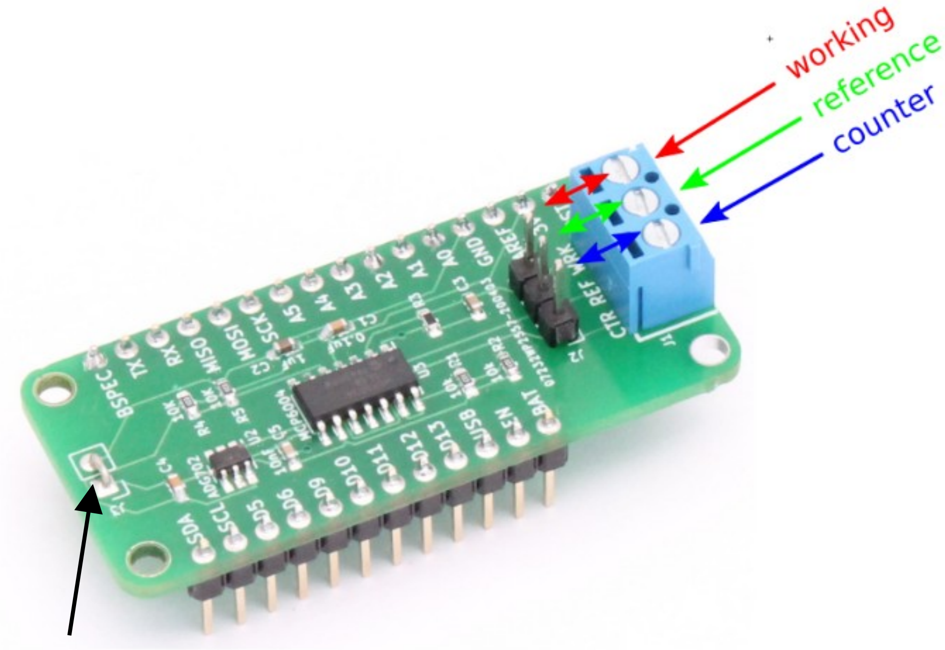
Virtual ground unity gain buffer

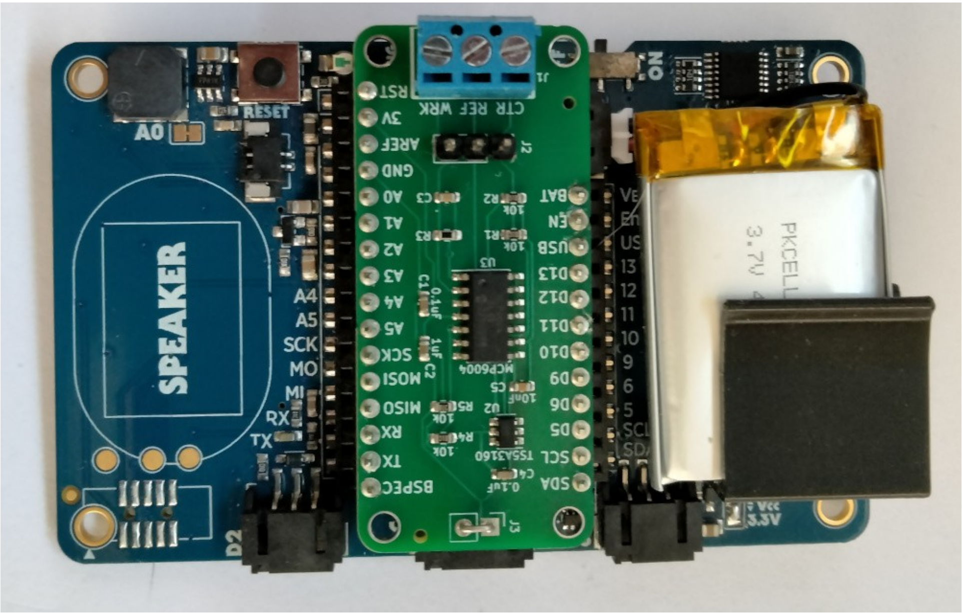
Screw terminal electrode connector

3−pin header electrode connector

Ground clip

PyBadge and Rodeostat Featherwing

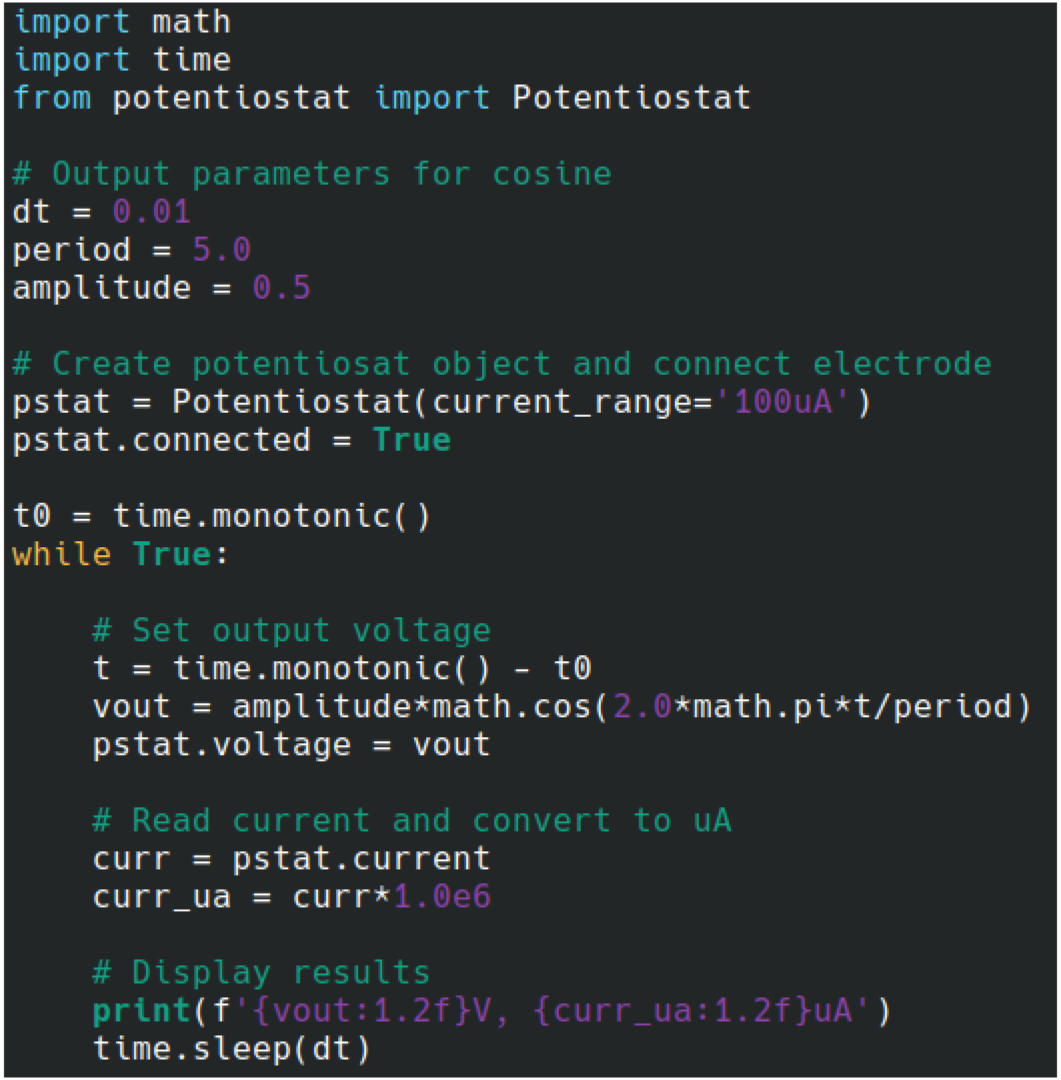




GND clip, 0V with respect to PyBadge not VGND

- for oscilloscope probes etc.

The Potentiostat class



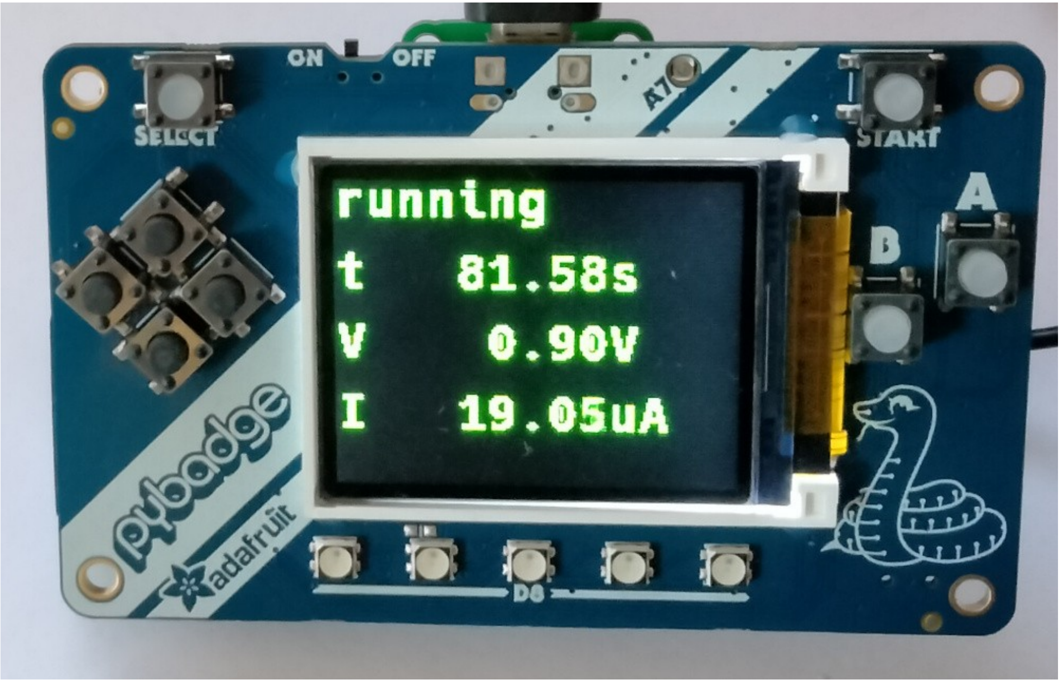
```python
import math
import time
from potentiostat import Potentiostat

# Output parameters for cosine
dt = 0.01
period = 5.0
amplitude = 0.5

# Create potentiosat object and connect electrode
pstat = Potentiostat(current_range='100uA')
pstat.connected = True

t0 = time.monotonic()
while True:

    # Set output voltage
    t = time.monotonic() - t0
    vout = amplitude*math.cos(2.0*math.pi*t/period)
    pstat.voltage = vout

    # Read current and convert to uA
    curr = pstat.current
    curr_ua = curr*1.0e6

    # Display results
    print(f'{vout:1.2f}V, {curr_ua:1.2f}uA')
    time.sleep(dt)
```

Potentiostat Examples

- **Cyclic voltammetry example with USB/serial between MCU and PC**

  - https://github.com/iorodeo/rodeostat_featherwing_example

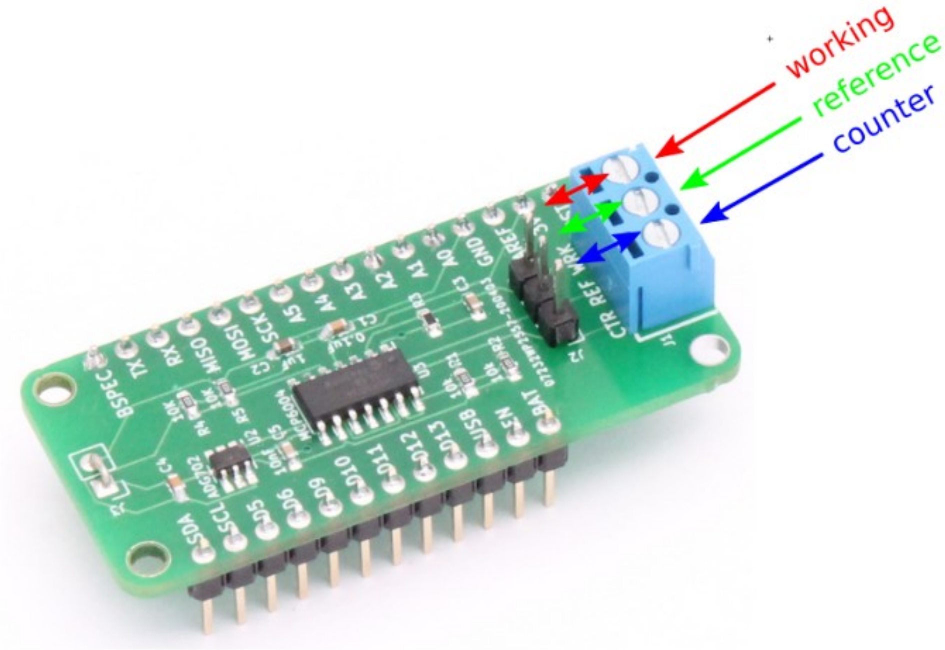
- **Stand-alone constant voltage example displaying time, voltage and current**



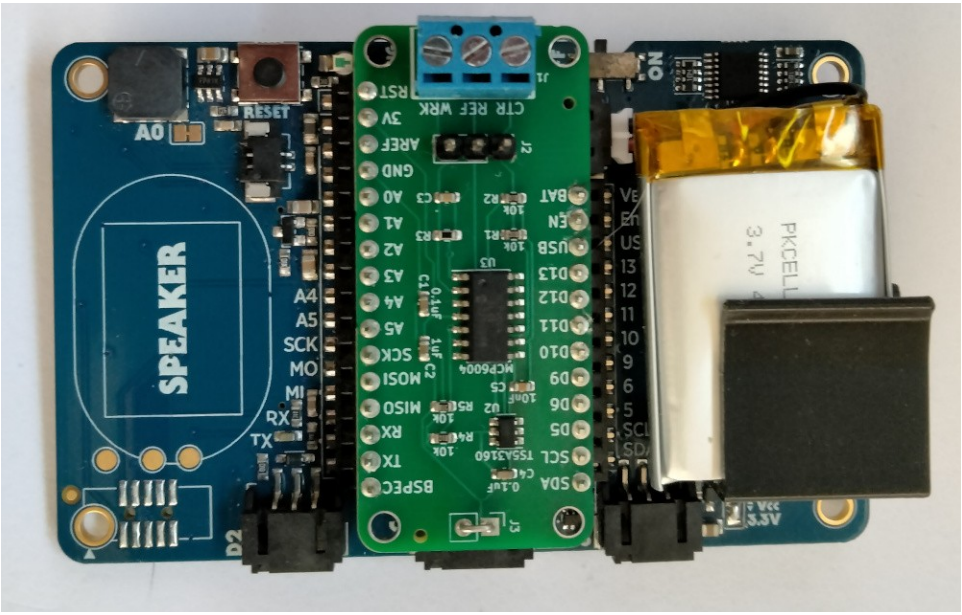
```
running
t    81.58s
V     0.90V
I   19.05uA
```

CirctuiPython Resources

- https://circuitpython.org/
- https://docs.circuitpython.org/en/latest/docs/index.html
- https://github.com/todbot/circuitpython-tricks
- https://circuitpython.org/awesome



working
reference
counter
CTR REF WRK



SPEAKER
RESET
CTR REF WRK
MCP6004
PKCELL
3.7V