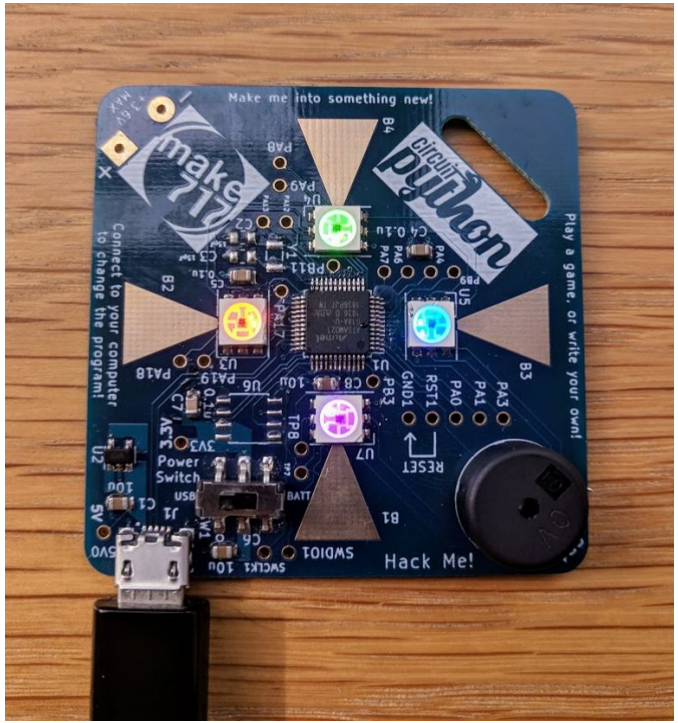
Controlling Hardware with CircuitPython

Snaking into your hardware

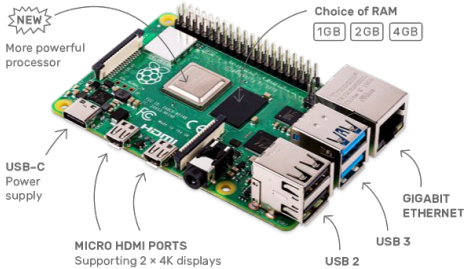
Hello

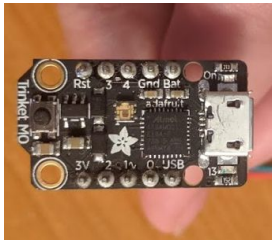
We've designed and manufactured open source custom printed circuit boards (PCBs) running CircuitPython

- Members at Make717 Innovation Center
  - Corey
    - Software Security Engineer
  - Bill
    - Hardware Engineer



Python on what?* (generalizations & blurry lines)





- Microprocessor (MP)
  - General Purpose
    - runs an operating system
    - multiple programs running
  - External RAM & memory storage
    - Typically Gigabytes
  - Examples: Intel 80486, Apple A8
  - Example Devices: Personal Computer, Laptop, Tablet, Raspberry Pi

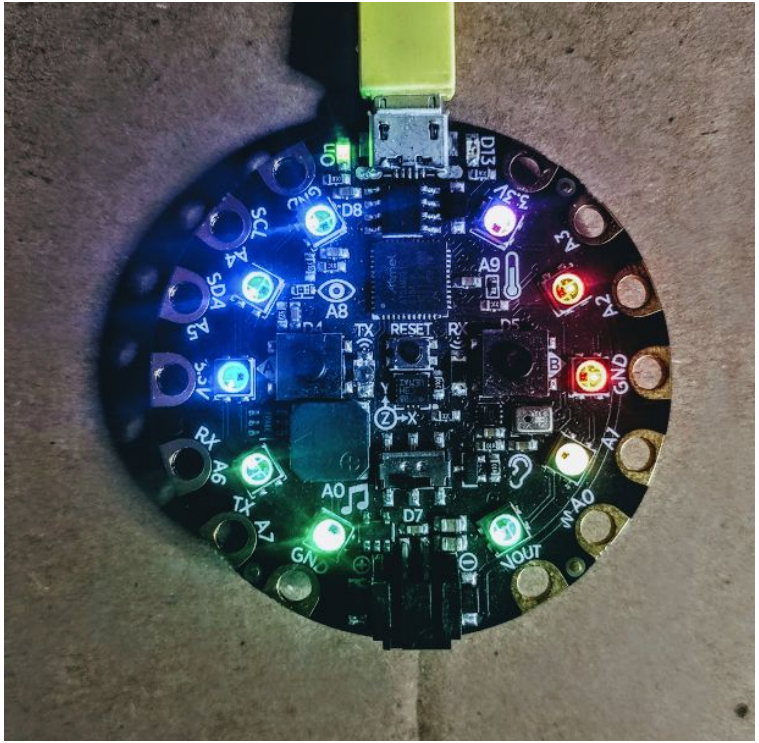
- Microcontroller (MCU)
  - Single Dedicated Chip
    - no operating system
    - single program running
    - access to all resources/memory
  - Built in RAM & ROM storage
    - Typically Kilobytes
  - Examples: ATmega328, SAMD21, STM32
  - Example Devices: Arduino, Teensy, Feather

https://www.raspberrypi.org/products/raspberry-pi-4-model-b/

What is CircuitPython

Implementation of Python 3.4 for Hardware

- Higher level programming language for MicroControllers
- Adafruit's port of MicroPython by Damien George
    - Adds native USB support
    - Designed for "beginners"
- Open Source
    - Software and libraries
    - A lot of hardware implementations



The REPL - DEMO

Read, Eval, Print and Loop

```
Press any key to enter the REPL. Use CTRL-D to reload.
Adafruit CircuitPython 4.1.0 on 2019-08-02; Adafruit CircuitPlayground Express with
samd21g18
>>> import board
>>> dir(board)
['__class__', 'A0', 'A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9',
'ACCELEROMETER_INTERRUPT', 'ACCELEROMETER_SCL', 'ACCELEROMETER_SDA', 'BUTTON_A',
'BUTTON_B', 'D0', 'D1', 'D10', 'D12', 'D13', 'D2', 'D3', 'D4', 'D5', 'D6', 'D7', 'D8',
'D9', 'I2C', 'IR_PROXIMITY', 'IR_RX', 'IR_TX', 'LIGHT', 'MICROPHONE_CLOCK',
'MICROPHONE_DATA', 'MISO', 'MOSI', 'NEOPIXEL', 'REMOTEIN', 'REMOTEOUT', 'RX', 'SCK', 'SCL',
'SDA', 'SLIDE_SWITCH', 'SPEAKER', 'SPEAKER_ENABLE', 'SPI', 'TEMPERATURE', 'TX', 'UART']
>>> import neopixel
>>> pixels = neopixel.NeoPixel(board.NEOPIXEL, 10)
>>> pixels[0] = (250, 0, 0)
>>>
```

Why Python

One of the fastest growing programming language

- Easier for beginners
  - No declaring variables
  - Strings are easy to work with
  - No memory mgmt (garbage collection)
- Popular for advanced topics
  - Data analytics
  - Machine Learning (ML) and Artificial Intelligence (AI)
- Large active developer community
  - Number of libraries and tutorials

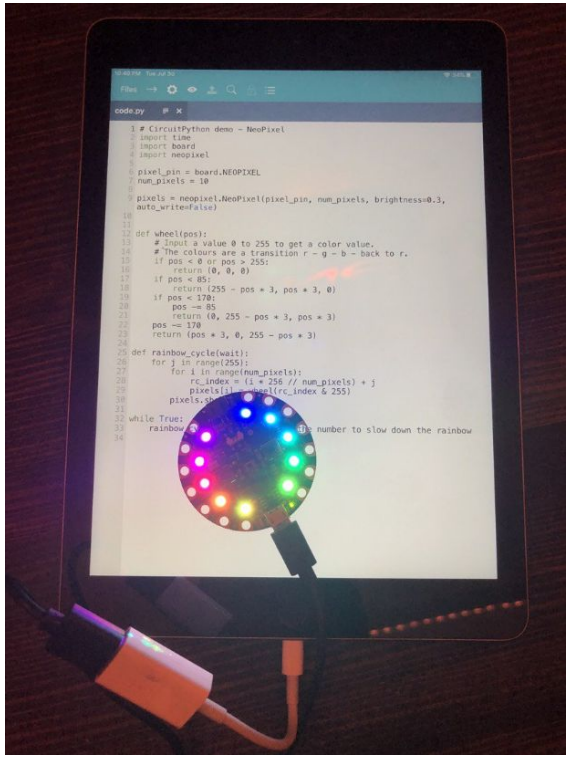
Why CircuitPython on Hardware

All those reasons and ...

- Fast iterations
    - No context switching to C
    - No compiling, easy to test line by line
    - Use Python code from other projects
- Preloaded libraries (on Express boards)
- Easy file storage

Why Python for Education

- Any computer can be the text editor
  - Chromebook support, working on iOS 13
- Source code is on the device
  - Don't need to distribute separately
- REPL allows for real-time changes
  - No wait compile to test loop
- Less cryptic error messages



https://blog.adafruit.com/2019/07/30

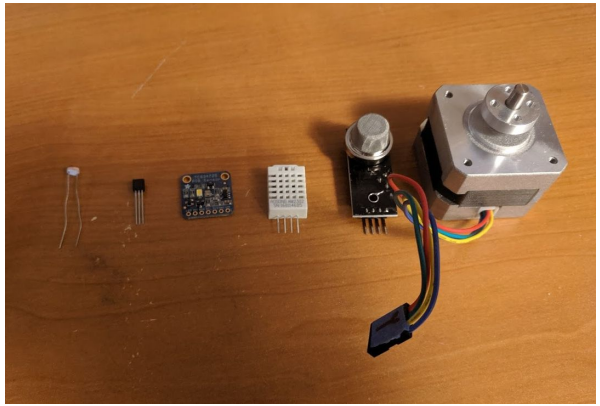
Error Messages

```
1  import board
2  import busio
3
4  REGISTERS = (0, 256)  # Range of registers to read, from the first up to (
5                        # not including!) the second value.
6
7  REGISTER_SIZE = 2     # Number of bytes to read from each register.
8
9  # Initialize and lock the I2C bus.
10 i2c = busio.I2C(board.SCL, board.SDA)
11 while not i2c.try_lock():
12     pass
13
```

Adafruit CircuitPython REPL

```
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Traceback (most recent call last):
  File "code.py", line 10, in <module>
RuntimeError: SDA or SCL needs a pull up
```

Libraries



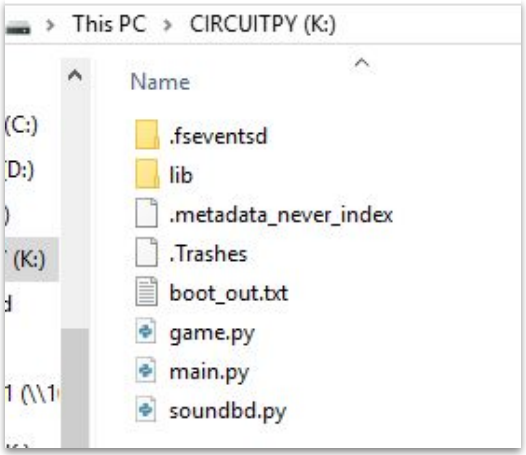
- Make it easy to use other people's sensors, components, or just code
- 172 library files currently in the Adafruit bundle
    - BLE (bluetooth), NeoPixel, RTC, servos, infrared, etc
- Library files are located in the /lib folder of the USB drive.
- Compiled Python code to MPY files
    - Small file size, optimized speed
    - You can download and "import" non-compiled as well

https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-libraries
https://github.com/adafruit/Adafruit_CircuitPython_Bundle/tree/master/libraries/drivers

Libraries

Raspberry Pi

- Setup: Enable i2c and SPI , then install RPI.GPIO
- Make sure to use Python3
    - `pip3 install adafruit-blinka`



Micro-Controller

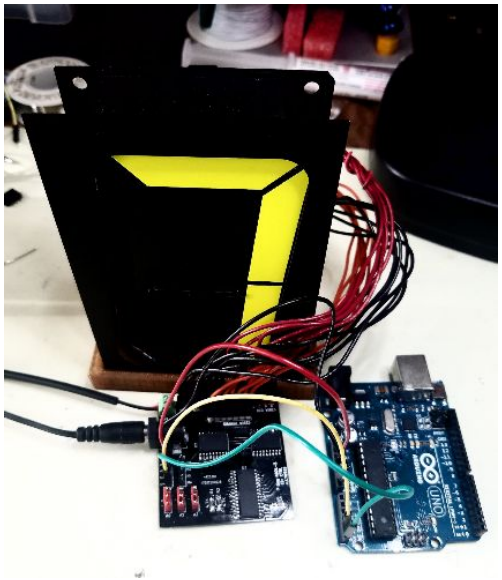
- Download package for correct CircuitPython version
    - https://circuitpython.org/libraries
- Unzip to "lib" folder on device (K:\CIRCUITPY\lib)

https://learn.adafruit.com/circuitpython-on-raspberrypi-linux/installing-circuitpython-on-raspberry-pi

Libraries

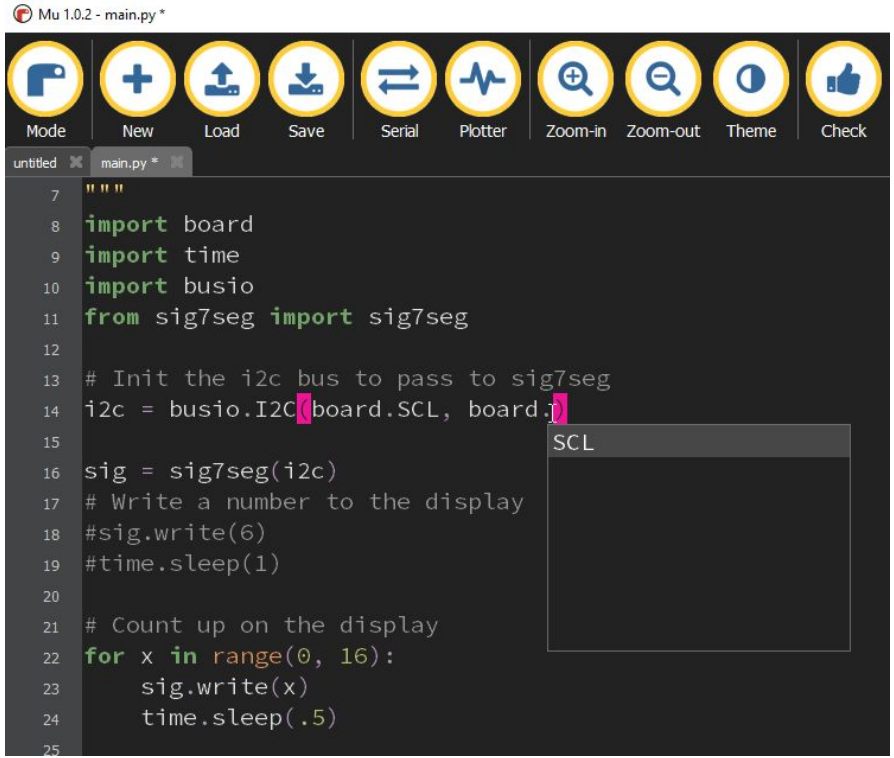
- Write Your Own
  - Make it easier for others to use your hardware or code
  - Test things out by including readable PY file (import)
- Compile to an MPY file
  - pip install mpy-cross
  - Use the "mpy_cross.py" file with CircuitPython source
  - TEST THE MPY FILE ON YOUR HARDWARE
    - Exact version of CircuitPython can matter

This can be a good place to compile in a challenge ;-)

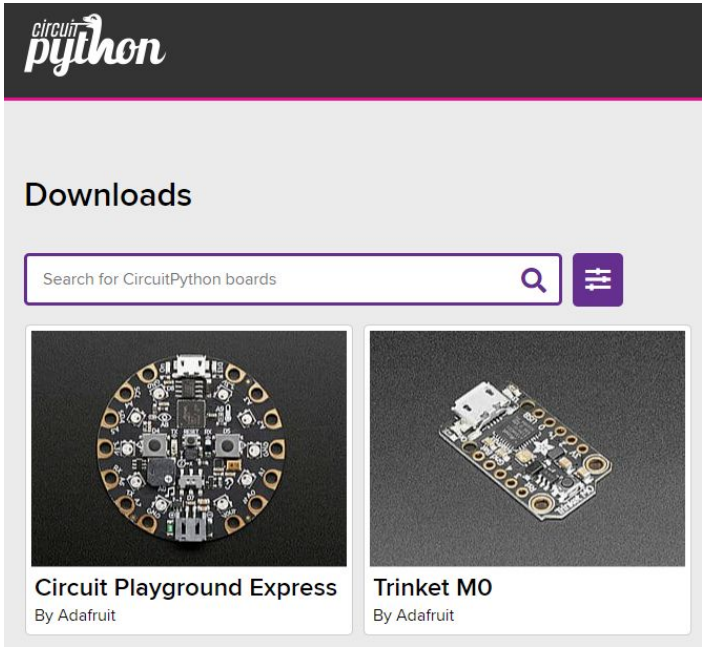


Python Editor

- Any will work, but Mu is nice
  - Simple, but highly functional
  - Micro & CircuitPython support (+more)
  - Code & REPL
  - Syntax highlighting
  - Code lint and format checking
  - Open Source
  - https://madewith.mu
  - https://codewith.mu/en/



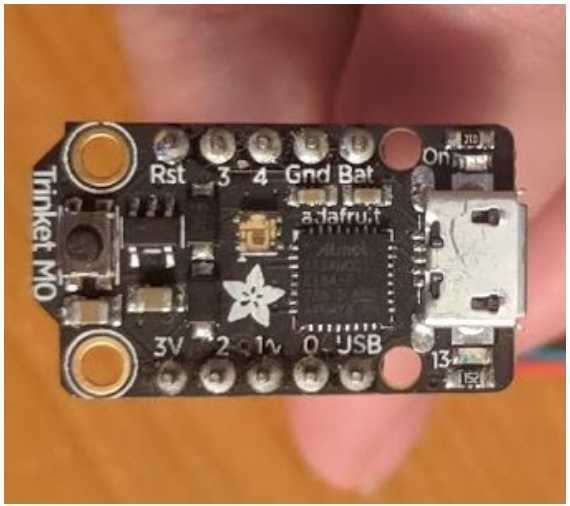
Commercial Boards



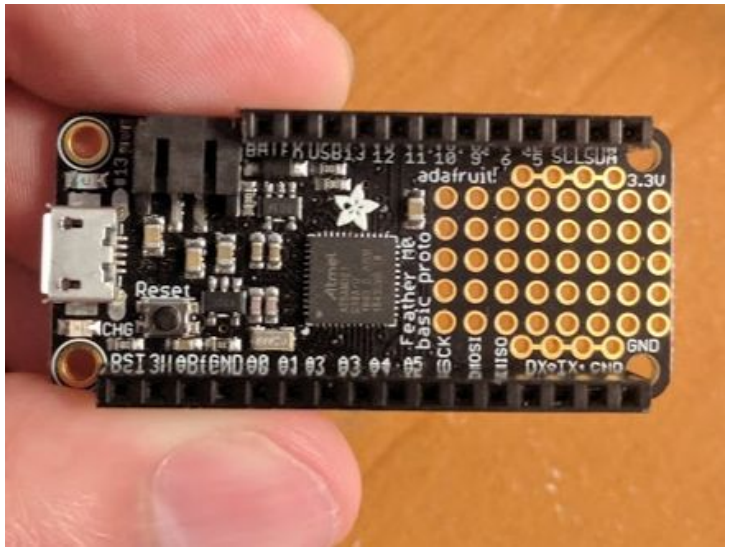
Good list at - **https://circuitpython.org/downloads**

- "M0" or "M4" boards are ARM Cortex
  - M4 is typically faster
- "Express" boards typically include 2MB SPI Flash
  - Enough room for all compiled libraries (MPY files)
  - You can remove this for more room for your own code
- "nRF528xx" boards have bluetooth radios
- Support from an number of manufactures
  - Adafruit, Nordic Semi, Arduino (zero), SparkFun, Particle, STMicroelectronics

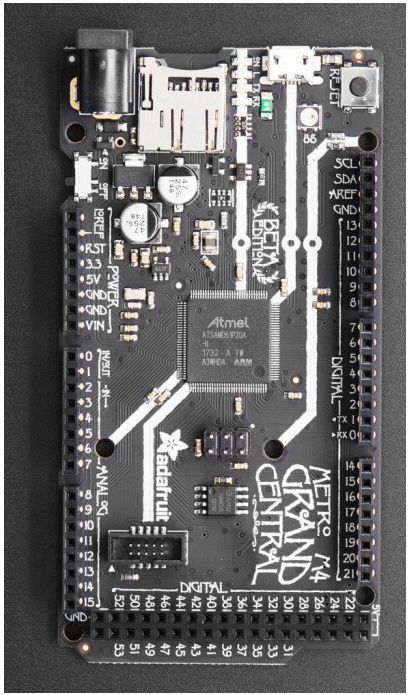
Popular Commercial Boards



Trinket M0
$8.95



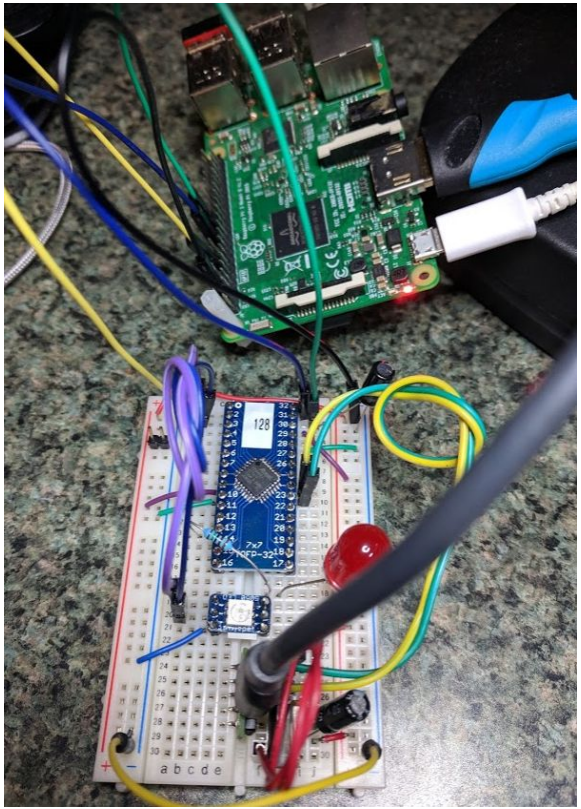
Feather M0
Basic $19.95



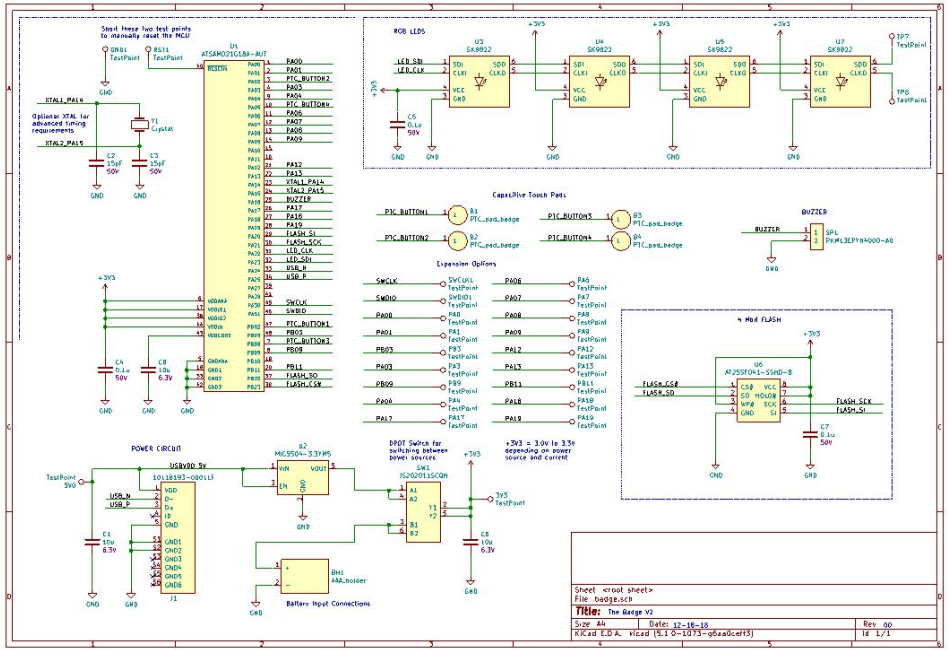
Grand Central
M4 $37.50

Build your own board

- Why should I do this?
  - Cost (Can be cheaper if building large quantities)
  - Form Factor (Wear your PCB!)
  - It's cool and fun (says people at the makerspace)
- Make it Easy
  - Use a processor that's currently supported
    - SAMD21, nRF52, STM32F4
    - RAM: 32KiB or more (CP takes about 16KiB)
    - ROM: 256KiB standard
      - smaller might require removing common functions
  - Prototype it first
    - Breakout boards for SMD parts



Build your own board - Easy as 1,2,3 :-)



1. Design the Schematic
2. Layout the PCB
3. Send files to Manufacturer

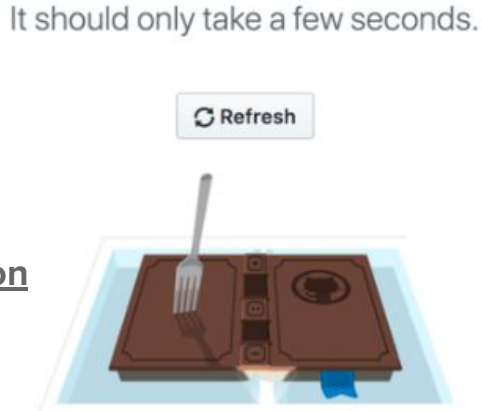
- Use Open Source Hardware Examples!
  - Search Github for Adafruit PCB designs
    - Schematics and Board files
    - Trinket M0 (SAMD21**E**)
    - Feather M0 Basic (SAMD21**G**)
  - Or our Make717 files :-)
    - in KiCad
    - https://github.com/make717gh

BoMs Away (our Bill of Materials)

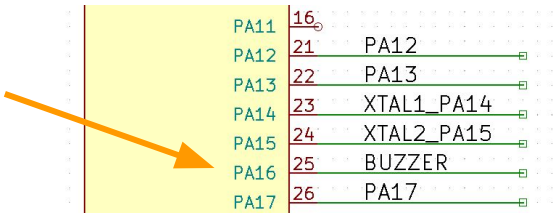
Designator	Quantity	Designation	MFG	MFG #	Distributor	Price	Total Price
U7,U3,U4,U5	4	SK9822	Colorful Pearl	SK9822	Ebay	$0.11	$0.44
J1	1	10118193-0001LF	Amphenol FCI	10118193-0001LF	Digi-Key	$0.30	$0.30
SW1	1	JS202011SCQN	Vimex	MSS22D18	Ebay	$0.13	$0.13
C1,C6,C8	3	10u	Samsung	CL21A106KQCLNNC	Digi-Key	$0.04	$0.12
C4,C5,C7	3	0.1u	Samsung	CL21F104ZBCNNNC	Digi-Key	$0.02	$0.06
U1	1	ATSAMD21G18A-AUT	Microchip	ATSAMD21G18A-AU	Digi-Key	$2.60	$2.60
SP1	1	PKM13EPYH4000-A0	Murata	PKM13EPYH4000-A0	Digi-Key	$0.34	$0.34
U2	1	MIC5504-3.3YM5-TR	Microchip	MIC5504-3.3YM5-TR	Digi-Key	$0.08	$0.08
BH1	1	Battery Holder	TrendBox	B071DK454L	Amazon	$0.40	$0.40
AAA Battery	2	Battery	Generic	AAA	Amazonhttps:/	$0.09	$0.19
Lanyard	1	N/A	N/A	N/A	Ebay	$0.08	$0.08
PCB	1	PCB with Stencil	Make717	badge_v2a	PCBWay	$0.80	$0.80
						Total	**$5.54**

$5.54 (each for 250)

Build your own board

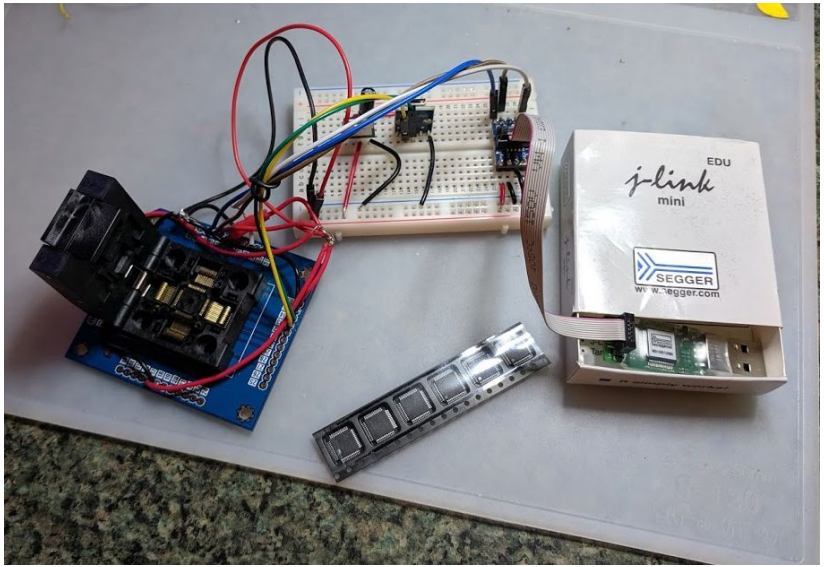


- Custom Build of CircuitPython
  - https://learn.adafruit.com/how-to-add-a-new-board-to-circuitpython
    - Fork the CircuitPython repo
    - Give your board a name!!!
    - Copy and Edit: `mpconfigboard.h, mpconfigboard.mk, pins.c`
  - Pins.c
    - This maps the MCU pins to the "board" object
    - { MP_ROM_QSTR(MP_QSTR_**BUZZER**), MP_ROM_PTR(&pin_**PA16)** },
  - mpconfigboard.h
    - Use IGNORE_PIN to have a smaller firmware file
    - Set the MICROPY_HW_LED_STATUS

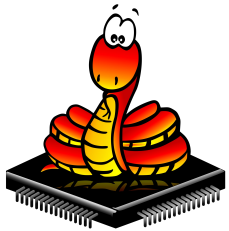


Build your own: Firmware

- Flashing your code to a blank chip
  - Programmer or OpenOCD on RPi
  - Watch Chip Voltage!!!
- Once installed, updates can be drag and dropped over USB drive
  - Double tap reset to enter bootloader mode (or short reset pin)
- Can customize boot in UF2
  - Drive name
  - blink LED or NeoPixel
  - https://github.com/adafruit/uf2-samdx1



Pythons: Micro vs Circuit



- MicroPython
  - Broader hardware support
    - "machine" file for accessing Pins
  - Libraries customized for hardware
    - utime, uos, ujson, etc..
    - pby, esp32 (specific MCUs)
  - Broader hardware support
    - ESP32, ESP8266, Teensy 3.x, SAMD21, Micro:Bit, K201 (RISC-V), etc...
- CircuitPython
  - Built for "Beginners"
    - "board" file for accessing Pins
    - Closer to CPython library naming
      - time module
  - Limited Microcontroller support
    - MCU USB support required
    - SAMD21, SAMD52, nRF, etc..

The Sacrifices

Less storage (~50KB)

Higher RAM requirements

Slightly longer boot

Hardware interrupts*

Low Power optimization*

Speed* (easily run 1/10 the speed)

http://www.quickmeme.com/meme/3s6kah

Questions?

Thank you to CPOSC and Make 717

- https://circuitpython.org
- https://micropython.org