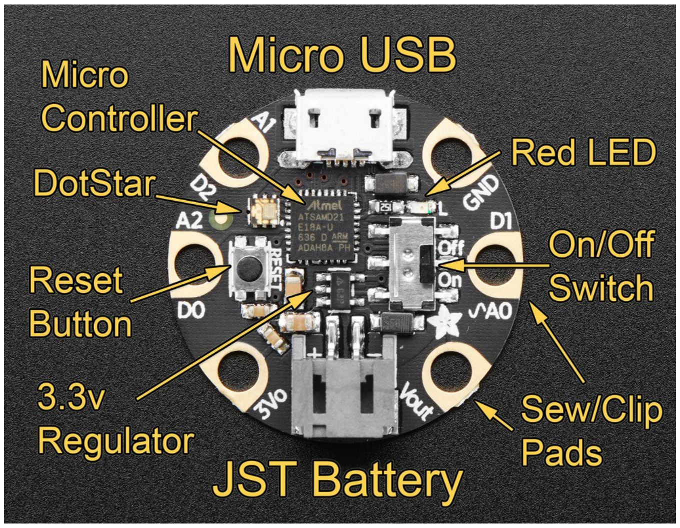
Adafruit Gemma M0 CircuitPython Quickstart



Your Adafruit Gemma M0 has CircuitPython on board! It's a MicroChip SAMD21 microcontroller running at 48 MHz, with 32kB RAM and 256kB flash: 192kB for CircuitPython, plus 64kB for USB drive **CIRCUITPY**.

Check out these Adafruit Learn Guide Links!
Welcome to CircuitPython: adafru.it/cpy-welcome
CircuitPython Essentials: adafru.it/cpy-essentials
Gemma Guide: learn.adafruit.com/adafruit-gemma-m0

Are you on Windows 7?
You need to install drivers before plugging in! See **Welcome->Installing CircuitPython**. Windows 10, Mac, and Linux don't need drivers. There's a Windows 7 driver right on **CIRCUITPY**, but prefer the downloaded version.

Plug It In!
Use a micro-USB cable with data (beware charge/power-only cables). A USB drive called **CIRCUITPY** will appear. If there's a **code.py** (or **main.py**) on **CIRCUITPY**, it will run automatically. There is a demo **code.py** on the board that you have. Check it out and make a copy of it, because it has lots of example code. And you'll find an https://adafruit.com discount code in the README in **CIRCUITPY**.

Avoiding Filesystem Corruption
Windows and Linux don't write back data to **CIRCUITPY** immediately: they can delay for 10s of seconds. (Not an issue on MacOS.) **Eject or sync after you copy files, and always before you unplug or press the Reset button.** Otherwise **CIRCUITPY** may become corrupted. Continue reading to see editors that write immediately so you don't need to Eject or sync every time you edit. If **CIRCUITPY** does get corrupted, see **Restoring CircuitPython and Initial CIRCUITPY** in this Quickstart.

Editing Code
Mu is the easiest editor to use: it includes a Python editor and easy REPL access. See **Welcome-> Installing Mu Editor**. The latest version for Windows is there. (The Mac version is old.) Or, for any OS, you can use GitHub and **pip** to install the latest version. (Use a virtualenv if you want.)

```
git clone https://github.com/mu-editor/mu
cd mu
pip install --user .
```

If you're using an older version of Mu, it may write some sounds and image files into **CIRCUITPY**, which fills up the filesystem. Delete the files in the **images** and **sounds** folders, but not the folders themselves, or else they'll be recreated.

If you don't use Mu, use an editor that writes back immediately: VS Code, Atom (install the circuitpython-force-to-drive package), Sublime, gedit, vim with `-n` option, emacs, PyCharm with Safe Write. ***Don't use*** Notepad, nano, IDLE. See **Welcome-> Creating and Editing Code**.

Auto-Reload
Every time you write a file, **code.py** (or **main.py**) will be re-run, unless you are in the REPL. Just edit **code.py** and see it run right away. This makes for a fast workflow.

Libraries
CircuitPython has builtin native libraries, but also has libraries written in Python (which are compiled into **.mpy** files to save space). Your board has a **lib** folder with some useful libraries, including the **adafruit_DotStar** and **neopixel** libraries mentioned below. If you don't have a lib folder, you can restore the board (see below)

Restoring CircuitPython and Initial CIRCUITPY
In learn.adafruit.com/adafruit-gemma-m0/downloads, you'll find **Gemma-CircuitPython-2.3.1-PyCon.uf2**. This file contains has an even newer version of CircuitPython than is on your Gemma M0, and also all the original files on **CIRCUITPY**. To update or restore your board, double-click the reset button. The red LED will pulse, and you'll see **GEMMABOOT** as a USB drive. Copy the **.uf2** file above to **GEMMABOOT**, wait a few seconds, and **CIRCUITPY** will reappear. ***Warning: any files you wrote to the board will go away!*** You can copy **CURRENT.UF2** from **GEMMABOOT** back to you computer to get a full backup of your Gemma, including everything in **CIRCUITPY**. (Doesn't work on Express boards like Circuit Playground Express.)

Talk to the REPL!
Connect to the REPL with Mu, or use **Putty** or **Tera Term** (Windows), or **screen** or **picocom** (Mac and Linux). Type Enter if necessary to start the REPL. If **code.py** is running, type ctrl-C. Type ctrl-D to soft-restart.

```
>>> 1+2
3
```

Blink - the “Hello World” of CircuitPython!
Make your Gemma red LED blink. Type this into the REPL or **code.py**:

```
import board, digitalio, time
led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
while True:
    # led.value is True or False
    led.value = not led.value
    time.sleep(0.5)
```

What’s the Temperature?

```
import microcontroller
# cpu on-chip sensor
print(microcontroller.cpu.temperature)
```

Capacitive Touch Detection!

```
import board, touchio, time
# works for A1 and A2 also
touch = touchio.TouchIn(board.A0)
while True:
    if touch.value:
        print("Touched!")
        time.sleep(0.05)  # debounce
```

Light up the DotStar On-Board RGB LED!

```
import time, board, adafruit_dotstar
# a DotStar strip of length 1
rgb = adafruit_dotstar.DotStar(
board.APA102_SCK, board.APA102_MOSI, 1)
rgb.brightness = 0.3 # range is 0-1.0
while True:
    for i in range(256):
        rgb[0] = (i, 0, 0) # (r,g,b):0-255
        time.sleep(0.01)
    for i in range(255, -1, -1):
        rgb[0] = (0, i//2, i)
        time.sleep(0.01)
```

Use External NeoPixels!
We’ll have some NeoPixel rings for you to try at our CircuitPython Open Spaces at Pycon.

```
import time, board, neopixel
ring = neopixel.NeoPixel(board.D1, 16,
auto_write=False)
ring.brightness = 0.1
def wheel(pos):
    if pos < 0 or pos > 255:
        return (0, 0, 0)
    if pos < 85:
        return (255 - pos * 3, pos * 3, 0)
    if pos < 170:
        pos -= 85
        return (0, 255 - pos * 3, pos * 3)
    pos -= 170
    return (pos * 3, 0, 255 - pos * 3)
def rainbow_cycle(delay):
    for j in range(0, 255, 8):
        for i in range(ring.n):
            idx = (i * 256 // ring.n) + j
            ring[i] = wheel(idx & 255)
        ring.show()
        time.sleep(delay)
while True:
    rainbow_cycle(0.001)
```

Read Analog Input!
We’ll have some potentiometers at our Open Spaces.

```
import time, board, analogio
a0 = analogio.AnalogIn(board.A0)
while True:
    print(a0.value)
    time.sleep(1)
```

Check out the Learn Guides mentioned at the front of this Quickstart for lots more examples, and also see the **code.py** *that comes on the board. Come visit the CircuitPython Open Spaces and see the Adafruit folks Monday at the Sprints. Have fun!*