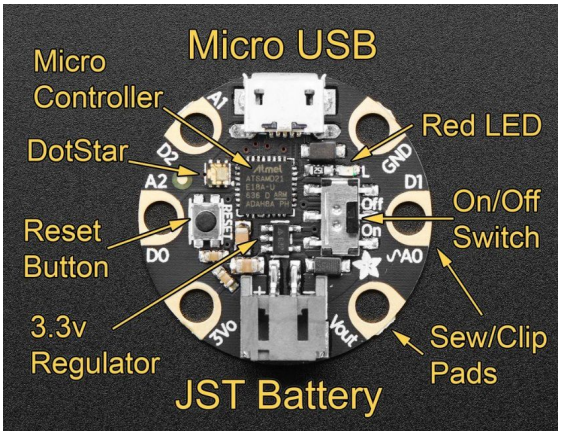
Guide de démarrage
Gemma M0 Adafruit - CircuitPython



Votre Gemma M0 Adafruit est livré avec CircuitPython !
C'est un microcontrôleur MicroChip SAMD21 cadencé à 48 MHz, avec 32 ko de RAM et 256 ko de flash : 192 ko pour CircuitPython, et 64 ko de stockage USB **CIRCUITPY**.

Ces guides (en anglais) sont à votre disposition
Welcome to CircuitPython : adafru.it/cpy-welcome
CircuitPython Essentials : adafru.it/cpy-essentials
Gemma Guide : learn.adafruit.com/adafruit-gemma-m0

Vous utilisez Windows 10, Linux ou MacOS ?
Pas besoin de driver. Sinon, consultez le guide "**Welcome**" ci-dessus.

Branchez-le !
Bon, brancher n'importe quoi que vous recevez à une conférence, ce n'est pas toujours une bonne idée, mais si vous vous nous faites confiance et que vous faites confiance à Adafruit, branchez votre carte via le port micro-USB (attention, certains câbles ne transmettent que la puissance, pas les données). Un disque USB nommé **CIRCUITPY** apparaîtra. Tout fichier nommé **code.py** ou **main.py** sur **CIRCUITPY** sera automatiquement lancé sur la carte. Il devrait y avoir un fichier **code.py** sur votre carte. Lisez-le et faites-en une copie, il contient plein d'exemples de code. Sans compter que le README sur le disque USB contient un code de réduction à utiliser sur https://adafruit.com.

Éviter la corruption du système de fichier
Windows et Linux n'écrivent pas les données sur **CIRCUITPY** immédiatement : il peut parfois s'écouler jusqu'à 10 secondes (cela n'arrive pas sous MacOS). Après avoir copié des fichiers sur la carte, éjectez-la ou synchronisez la systématiquement avant de la débrancher ou d'appuyer sur le bouton reset, ce sans quoi **CIRCUITPY** pourrait être corrompu. Un peu plus bas, vous trouverez des éditeurs de texte qui synchronisent proprement leurs données, ce qui vous permettra de ne pas avoir à éjecter ou synchroniser à chaque édition. Si **CIRCUITPY** est tout de même corrompu, lisez la section "Restauration" de ce guide.

Modifiez le code
Mu est un éditeur de texte particulièrement simple : il comprend des fonctionnalités Python et un accès au shell Python. Regardez le guide **Welcome** ou directement codewith.mu pour des instructions d'installation détaillées. Vous pouvez l'installer simplement sur n'importe quel ordinateur via **pip** (et éventuellement un virtualenv) :

```
pip install mu-editor
```

Bien sûr, tous les éditeurs classiques marcheront, et ne causeront pas de corruption : VS Code, Atom (installez l'extension circuitpython-force-to-drive), Sublime, gedit, vim avec l'option -n, emacs, PyCharm avec "Safe Write".
N'utilisez pas Notepad, nano, ou IDLE.

Auto-Reload
À chaque fois que vous sauvegardez un fichier, code.py (ou main.py) sera relancé, sauf si vous êtes sur le REPL. Sauvez, rechargez, sauvez rechargez, itérez, hackez, c'est parti :)

Libraires
CircuitPython contient la librairie standard, mais aussi des librairies Python (compilées en **.mpy** pour gagner de la place). Votre carte contient un dossier **lib** avec quelques librairies utiles, comme **adafruit_DotStar** et **neopixel** (voir ci-après). Si le dossier lib n'est pas présent, restaurez la carte (voir ci-dessous).

Restaurer CircuitPython et CIRCUITPY
Attention : tout le contenu de votre carte disparaîtra !
À l'adresse
learn.adafruit.com/adafruit-gemma-m0/downloads
vous trouverez
Gemma-CircuitPython-2.3.1-PyCon.uf2.
Ce fichier contient la version à jour de CircuitPython et tous les fichiers initiaux de **CIRCUITPY**. Pour mettre à jour ou restaurer votre carte, double-cliquez sur le bouton **reset**. La LED rouge clignotera et vous verrez un disque USB nommé **GEMMABOOT**. Copiez le fichier **.uf2** sur

GEMMABOOT, attendez quelques secondes et **CIRCUITPY** réapparaîtra. Vous pouvez copier **CURRENT.UF2** de **GEMMABOOT** pour backuper l'ensemble de votre carte sur votre ordinateur, incluant **CIRCUITPY**.

Parlez au REPL !

Connectez-vous au REPL (shell python) via Mu, ou via **Putty** ou **Tera** (Windows) ou **screen** ou **picocom** (Mac / Linux). Appuyez sur entrée si nécessaire pour démarrer le REPL. Si **code.py** tourne, appuyez sur ctrl+C. Appuyez sur ctrl+D pour redémarrer

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

Blink - le "Hello World" de CircuitPython !

Faisons clignoter la LED rouge du Gemma. Voici le code à taper dans le REPL ou dans **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)
```

Il fait combien ?

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

Détection capacitive du toucher !

```
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
```

Allumer la LED RGB DotStar intégrée !

```
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 reverse(range(256)):
        rgb[0] = (0, i//2, i)
        time.sleep(0.01)
```

Allumer des Neopixels ou utiliser des entrées analogiques

Bon, on n'en a pas à vous proposer ici, mais si vous en avez, voilà des exemples de code pour interagir avec des NeoPixels ou des potentiomètres :

NeoPixel

```
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)
```

Potentiomètre

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

Jetez un oeil aux guides mentionnés au début de ce document pour plus d'exemples. Regardez aussi le ***code.py*** *livré avec la carte et visitez le site web d'Adafruit. Amusez-vous bien !*