



¡Bienvenido a CircuitPython!

Created by Kattni Rembor



<https://learn.adafruit.com/bienvenido-a-circuitpython-2>

Last updated on 2024-10-31 02:52:32 PM EDT

Table of Contents

Resumen	5
<ul style="list-style-type: none">• ¡Esta guía te ayudara a iniciar con CircuitPython!	
¿Que es CircuitPython?	5
<ul style="list-style-type: none">• CircuitPython está basado en Python• ¿Por qué debería utilizar CircuitPython?	
Preguntas frecuentes	7
<ul style="list-style-type: none">• He venido utilizando CircuitPython 3.x o 2.x, ¿donde puedo encontrar librerías compatibles?• ¿Que es MemoryError?• ¿Que hago cuando recibo un MemoryError?• ¿Puede el orden de mis import, afectar la memoria?• ¿Como puedo crear mis propios archivos .mpy?• ¿Como reviso cuanta memoria tengo disponible?• ¿CircuitPython maneja interrupciones?• ¿Las Feather M0 trabajan con WINC1500?• ¿Pueden chips AVR como los ATmega328 o ATmega2560 correr CircuitPython?• Acrónimos Comunes	
Instalando el editor Mu	10
<ul style="list-style-type: none">• Descargando e instalando Mu• Usando Mu	
Instalando CircuitPython	12
<ul style="list-style-type: none">• ¡Descarga la última versión!• Controladores para Windows 7• Iniciando el gestor de arranque UF2• ¿Cual es la diferencia entre CIRCUITPY y nombretarjetaBOOT?• Nombres de unidades de disco en modo gestor de arranque	
La unidad de disco CIRCUITPY	16
<ul style="list-style-type: none">• Trabajando con múltiples dispositivos• Renombrando CIRCUITPY• Renombrando CIRCUITPY en Mac• Renombrando CIRCUITPY en Windows• Renombrando CIRCUITPY en Linux• Renombrando CIRCUITPY desde CircuitPython• Revirtiendo a CIRCUITPY	
Creando y editando código	21
<ul style="list-style-type: none">• Creando código• Editando código• 1. Utiliza un editor que escriba los archivos completamente cuando los salvas.• 2. Expulse o sincronice el dispositivo luego de escribir• Volviendo con la edición de código...• Explorando tu primer programa en CircuitPython• Imports y librerías• Configurando el LED• Ciclos• Más Cambios• Nombrando tu programa	

Conectándose a la Consola Serial	29
<ul style="list-style-type: none">• ¿Estás utilizando Mu?• Configurando permisos en Linux• ¿Utilizando otra cosa?	
Interactuando con la Consola Serial	31
El REPL	34
<ul style="list-style-type: none">• Retornando a la consola serial	
Hardware para CircuitPython	39
<ul style="list-style-type: none">• Trinket M0• Gemma M0• Circuit Playground Express• Feather M0 Express• Metro M0 Express• ¿Que Sigue?	
Librerías para CircuitPython	45
<ul style="list-style-type: none">• Instalando el Agrupado de Librerías de CircuitPython• Archivos Ejemplo• Copiando librerías a tu tarjeta• Ejemplo: ImportError debido a librerías faltantes• Instalación de librerías en tarjetas que no son Express• Actualizando Librerías o Ejemplos de CircuitPython	
¡Bienvenido a la Comunidad!	50
<ul style="list-style-type: none">• Canal de Discord de Adafruit• Foros de Adafruit• Repositorios de Adafruit en GitHub• ReadTheDocs	
Consola Serial Avanzada en Windows	55
<ul style="list-style-type: none">• Controlador para Windows 7• ¿Que es el COM?• Instalando Putty	
Consola Serial Avanzada en Mac y Linux	59
<ul style="list-style-type: none">• ¿Cual es el puerto?• Conectándose con screen• Permisos en Linux	
PyCharm y CircuitPython	64
CircuitPython para la ESP8266	70
<ul style="list-style-type: none">• ¿Porqué no damos mantenimiento a las ESP8266?• Sobre ESP8266 para CircuitPython (3.x)• Instalando CircuitPython en una ESP8266• Descargue esptool• Descargue la última versión del firmware de CircuitPython• Alistando la ESP8266 para booteo• Borrar la ESP8266• Programar la ESP8266• ¡Suba librerías y archivos usando Ampy!• ¡Otras cosas para tomar en cuenta!	

Desinstalando CircuitPython

75

- Respalda tu Código
- Pasando a la Circuit Playground Express a MakeCode
- Pasándose a Arduino

Instalación sin UF2

78

- Subiendo con Bossac – Para las Feather M0 no-Express y Arduino Zero
- ¡Vamos a la línea de comando!
- Descarga el último firmware CircuitPython
- Descargando BOSSA
- Pruebe bossac
- Selección de puerto para Mac OS
- Entra en el gestor de arranque
- Ejecutando el comando bossac

Depuración

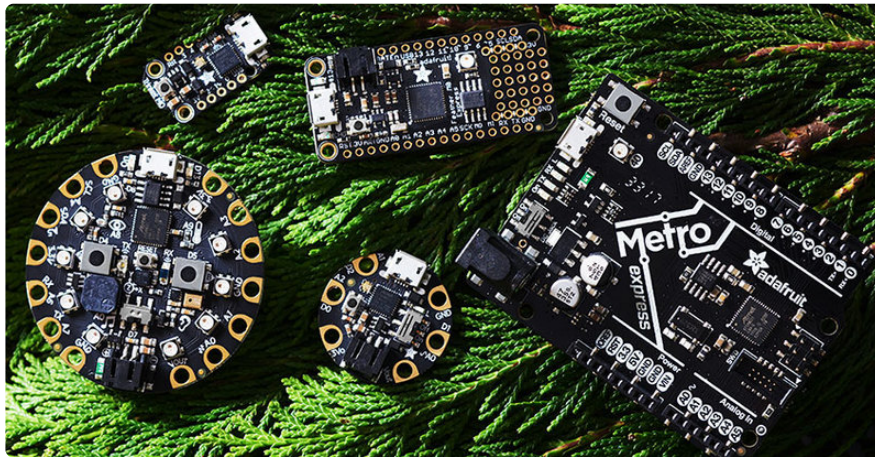
83

- Siempre Ejecuta la Última Versión de CircuitPython y sus Librerías
- Tengo que seguir usando CircuitPython 3.x o 2.x, ¿dónde puedo encontrar librerías compatibles?
- Unidades de disco CPLAYBOOT, TRINKETBOOT, FEATHERBOOT, or GEMMABOOT no presentes.
- Windows Explorer se queda pegado al acceder la unidad de disco nombretarjetaBOOT
- Copiar un UF2 a la unidad de disco nombretarjetaBOOT se queda pegado con un 0% de copia completada
- La Unidad de Disco CIRCUITPY no Aparece
- La Consola Serial de Mu no Imprime Nada
- Luz de Estado RGB para CircuitPython
- ValueError: Incompatible .mpy file.
- Problemas con la Unidad de Disco CIRCUITPY
- Forma sencilla: Usando `storage.erase_filesystem()`
- Forma antigua: Para las Circuit Playground Express, Feather M0 Express, y Metro M0 Express:
- Forma antigua: Para tarjetas no-Express con gestor de arranque UF2 (Gemma M0, Trinket M0):
- Forma antigua: Para tarjetas no-Express sin gestor de arranque UF2 (Feather M0 Basic Proto, Feather Adalogger, Arduino Zero):
- Quedándose Sin Espacio en Tarjetas No-Express

Esenciales para CircuitPython

96

Resumen



Entonces, tienes una nueva **tarjeta compatible con CircuitPython**. Las has conectado. Tal vez aparece como una unidad de disco llamada CIRCUITPY. ¡O tal vez no lo hizo!. De cualquier forma, es importante que sepas que hacer en este momento. Y bueno, ¡estamos para ayudarte!

¡Esta guía te ayudara a iniciar con CircuitPython!

Hay muchas cosas impresionantes sobre su nueva tarjeta. Una de ellas es su habilidad de correr CircuitPython. Es posible que hayas visto este nombre en algún lado del sitio de Adafruit. ¿No estás seguro de que es? ¡Nosotros te podemos ayudar!

"Pero yo nunca he programado en mi vida. ¡No hay forma de que lo logre!". ¡Absolutamente puedes! CircuitPython es diseñado para ayudarte a que aprendas desde las bases. Si eres nuevo a todo, ¡este es el lugar para empezar!

Esta guía te llevará por el camino para iniciarse con CircuitPython. Aprenderás como instalar CircuitPython, obtener la última versión de CircuitPython, como configurar una conexión serial, y como editar los archivos.

¡Bienvenido a CircuitPython!

¿Que es CircuitPython?

CircuitPython es un lenguaje de programación diseñado para simplificar la experimentación y aprendizaje de programar en microcontroladoras de bajo costo. Hace el iniciar más sencillo que nunca sin necesidad previa de descargar herramientas a la estación de trabajo. Una vez que tu tarjeta ha sido preparada, abres cualquier editor de texto, y puedes comenzar a escribir código. Es así de simple.



CircuitPython está basado en Python

Python es el lenguaje de programación de mayor crecimiento. Es enseñado en escuela y universidades. Es un lenguaje de programación de alto nivel, lo que significa que está diseñado para ser fácil de leer, escribir y mantener. Se utilizan módulos y paquetes lo que quiere decir que es sencillo reutilizar tu código en otros proyectos. Tiene un intérprete integrado, que significa que no hay pasos adicionales como compilación para que tu código funcione. Y por supuesto, Python es software Open Source, que quiere decir que es libre para ser usado, modificado o mejorado por cualquier persona. CircuitPython le agrega al hardware todas estas capacidades asombrosas. Si ya tienes conocimiento de Python, puedes aplicarlo usando CircuitPython. Si no tienes experiencia previa, ¡debería ser sencillo comenzar!



¿Por qué debería utilizar CircuitPython?

CircuitPython es diseñado para correr en tarjetas con microcontroladora. Una tarjeta con microcontroladora es una tarjeta con un chip controlador que la vuelve esencialmente una mini computadora todo-en-uno. ¡La tarjeta que estás sosteniendo es una tarjeta con microcontroladora! CircuitPython es sencillo de usar, dado que todo lo que necesitas es esta pequeña tarjeta, un cable USB, y una computadora con conexión USB. Pero eso es solo el principio.

Otras razones para usar CircuitPython incluyen:

- **Quieres tener tu ambiente listo para trabajar, en poco tiempo.** Creas un archivo, editas tu código, salvas el archivo y se ejecuta de inmediato. No hay proceso de compilado, sin descargas ni subidas necesarias.

- **Eres nuevo en programación.** CircuitPython es diseñado pensando en educación. Es sencillo comenzar a programar, y recibes retroalimentación inmediata de la tarjeta.
- **Actualiza sencillamente tu código.** Dado que tu código vive en la unidad de disco, puedes editarlo cuando gustes, y puedes mantener varios archivos con diferentes archivos para fácil experimentación.
- **La consola serial y REPL.** Estas te permiten recibir retroalimentación en vivo de tu código y programando de forma interactiva.
- **Almacenamiento de archivos.** El almacenamiento interno de CircuitPython es un excelente lugar para bitácoras de datos, tocar archivos de audio o para interactuar con archivos.
- **Fuertes capacidades de hardware.** Hay muchas librerías y controladores para sensores, tarjetas específicas y otros componentes externos.
- **¡Es Python!** Python es el lenguaje de programación de mayor crecimiento. Es enseñado en escuelas y universidades. CircuitPython es casi compatible con Python. Solamente agregar capacidades de hardware.

Esto es solo el comienzo. CircuitPython continua evolucionando, y actualizándose constantemente. Te damos la bienvenida y bien recibimos comentarios de la comunidad, y los incorporamos en como desarrollamos CircuitPython. Ese es el núcleo del concepto de Open Source. Esto mejora CircuitPython para tí y cualquier otro usuario.

Preguntas frecuentes

Estas son algunas de las preguntas comunes sobre CircuitPython y microcontroladoras con CircuitPython

Mientras continuamos desarrollando CircuitPython y crear nuevas versiones de producto, dejaremos de dar mantenimiento a versiones anteriores. Visite <https://circuitpython.org/downloads> para descargar la última versión de CircuitPython para tu tarjeta. Usted debe descargar el conjunto de librerías para CircuitPython (o "CircuitPython Library Bundle") para la versión apropiada de CircuitPython. Favor actualice CircuitPython y visite <https://circuitpython.org/libraries> para descargas la última versión del Conjunto de Librerías.

He venido utilizando CircuitPython 3.x o 2.x, ¿donde puedo encontrar librerías compatibles?

Ya no estamos compilando o dando mantenimiento a CircuitPython 2.x y 3.x. Lo alentamos a [actualizar CircuitPython a la última versión \(\)](#) y a utilizar una [versión](#)

[actualizada de las librerías.](#) () Sin embargo, puedes encontrar la [última versión disponible para 2.x compilada aquí](#) () y [la última para la versión 3.x aquí](#) ().



¿Puedo usar los ESP8266 o las ESP32 con CircuitPython? ¿Por qué no?

Estamos quitando capacidades para ESP8266 desde 4.x - ¡Para más información favor lea aquí!

<https://learn.adafruit.com/welcome-to-circuitpython/circuitpython-for-esp8266> (<https://adafru.it/CiG>)



¿Como me conecto a la Internet con CircuitPython?

Si desea agregar capacidades de Wifi, revise nuestra guía en utilizar ESP32/ESP8266 como un co-procesador. (<https://adafru.it/Dwa>)



¿Existen capacidades de asyncio en CircuitPython?

Nosotros no tenemos capacidades de asyncio en CircuitPython en este momento.



Mi LED Neopixel/DotStar de colores RGB parpadea en colores extraños. ¿Esto que significa?

El LED indicador de estado te puede decir que está pasando con tu tarjeta CircuitPython. ¡[Lea aquí lo que significan estos colores!](https://adafru.it/Den) (<https://adafru.it/Den>)

¿Que es `MemoryError`?

Los errores de solicitud de memoria suceden cuando tratamos de guardar mucho en la tarjeta. Las tarjetas con microcontroladoras para CircuitPython tiene una cantidad limitada de memoria disponible. Usted puede tener como 250 líneas de código en tarjetas MO Express. Si tratas de hacerle `import` muchas librerías, una combinación de librerías largas, o correr un código con muchas líneas de código, si programa va a fallar en ejecutar y vas a recibir un mensaje `MemoryError` en la consola serial (REPL).

¿Que hago cuando recibo un `MemoryError`?

Trata reiniciando tu tarjeta. Cada vez que reinicias la tarjeta, ella tratará de acomodar la memoria. Mientras es poco probable que resuelva tu error, es un paso sencillo que vale la pena probar.

Asegúrate de usar las versiones `.mpy` de las librerías. Todas las librerías de CircuitPython están disponibles en el conjunto en formato `.mpy` el cual consume menos memoria que el formato `.py`. Asegúrate de estar usando la [última versión de la librería \(\)](#) para tu versión de CircuitPython.

Si esto no resuelve tu problema, trata recortando tu código. Recortar comentarios, limpiar código innecesario o cualquier otra limpieza que puedas realizar para acortar tu código. Si estás utilizando muchas funciones, trata de moverlas a una librería separada, crear un `.mpy` de dicha librería e importándola en tu código.

Puedes transformar tu archivo entero a `.mpy` y hacer `import` en tu archivo `code.py`. Esto significa que no vas a poder editar tu código en vivo en la tarjeta,, pero te puede ahorrar espacio.

¿Puede el orden de mis `import`, afectar la memoria?

Si puede, dado que la memoria se fragmenta de diferente forma, dependiendo de el orden de solicitud y del tamaño de los objetos. Cargar archivos `.mpy` utiliza menos memoria por lo que se recomienda realizar esta conversión para archivos que no estés editando.

¿Como puedo crear mis propios archivos `.mpy`?

Puedes realizar tus propias versiones de archivos `.mpy` con `mpy-cross`.

Puedes descargar la versión de `mpy-cross` para CircuitPython 2.x para tu sistema operativo desde la página de [Versiones de CircuitPython \(\)](#) bajo la última versión para 2.x.

Usted puede compilar `mpy-cross` para CircuitPython 3.x, clonando el [repositorio de GitHub de CircuitPython \(\)](#) y ejecutando `make` dentro del directorio `circuitpython/`

`mpy-cross/`. Ahora ejecutas `./mpy-cross ruta/a/foo.py` para crear `foo.mpy` en el mismo directorio que el archivo original.

¿Como reviso cuanta memoria tengo disponible?

```
import gc
gc.mem_free()
```

Esto te dará el número de bytes disponibles para utilizarse.

¿CircuitPython maneja interrupciones?

No. CircuitPython no trabaja por el momento con interrupciones. No tenemos un estimado para cuando se podrían incluir.

¿Las Feather M0 trabajan con WINC1500?

No, la librería para WINC1500 no cabe en la memoria de flash de los chips M0.

¿Pueden chips AVR como los ATmega328 o ATmega2560 correr CircuitPython?

No.

Acrónimos Comunes

CP o CPy = [CircuitPython](https://adafru.it/cpy-welcome) (<https://adafru.it/cpy-welcome>)

CPC = [Circuit Playground Classic](http://adafru.it/3000) (<http://adafru.it/3000>)

CPX = [Circuit Playground Express](http://adafru.it/3333) (<http://adafru.it/3333>)

CPB = [Circuit Playground Bluefruit](http://adafru.it/4333) (<http://adafru.it/4333>)

Instalando el editor Mu

Mu es un simple editor de código que funciona con las tarjetas CircuitPython de Adafruit. Está escrito en Python y trabaja en Windows, MacOS, Linux y en Raspberry Pi. ¡La consola serial está integrada por lo que obtienes retroalimentación inmediata de la salida serial de tu tarjeta!

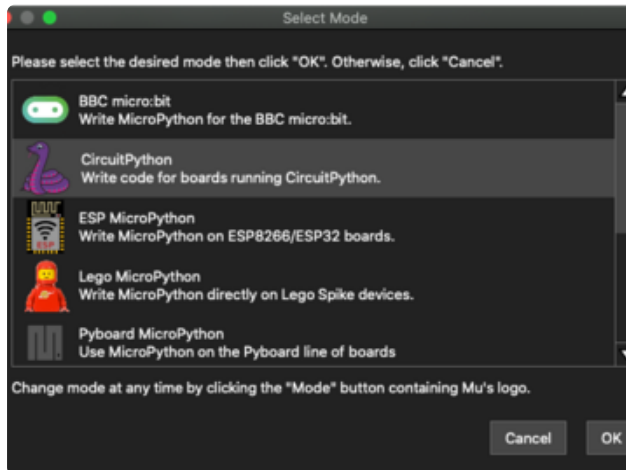
Mu es nuestro editor recomendado - favor úselo (¡a menos que seas un usuario experto y ya tengas un editor favorito!)

Descargando e instalando Mu



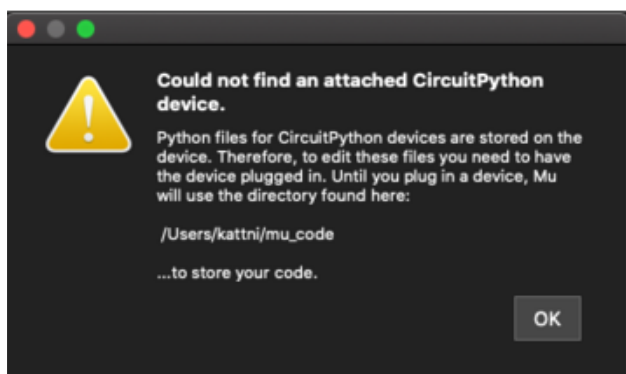
Descargue Mu de <https://codewith.mu> (). Presione los enlaces ya sea de **Download** o de **Start here** para descargas, e instrucciones de instalación. El sitio cuenta con valiosa información, incluyendo extensivos tutoriales y guías-como.

Usando Mu



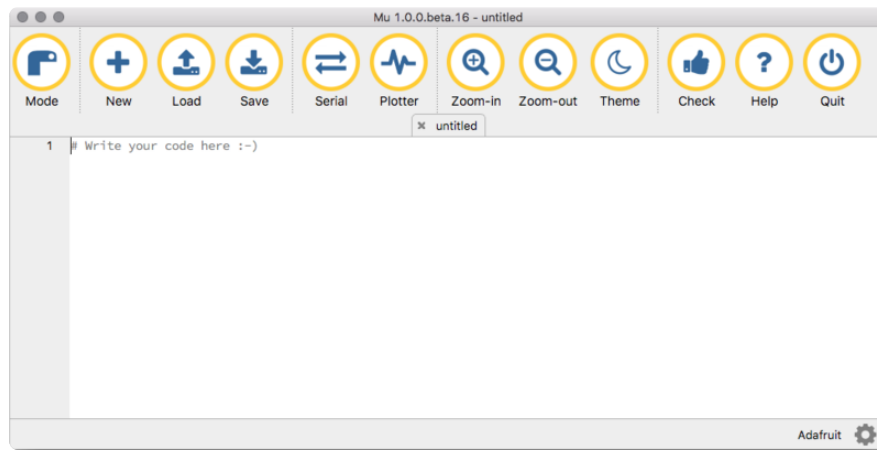
La primera vez que inicias Mu, te va a pedir que selecciones un modo (o "mode"), el cual puedes cambiar de nuevo en el futuro. ¡Por el momento selecciona **Adafruit!**

El modo actual se muestra en la esquina inferior izquierda de la ventana, a la par de un ícono de engranaje. Si el modo dice "Microbit" o algo similar, haga click en el para cambiar el modo, y escoja "Adafruit" de la ventana que aparece.



Mu va a tratar de auto-detectar tu tarjeta, así que conecta tu dispositivo CircuitPython y asegúrate de que aparezca como una unidad llamada CIRCUITPY, antes de iniciar Mu.

¡Ya estás listo para programar! Continuemos...



Instalando CircuitPython

Mientras continuamos desarrollando CircuitPython y creando nuevas versiones, vamos a ir descontinuando versiones anteriores. Si estás corriendo CircuitPython 2.x, necesitas actualizarte a 3.x. Normalmente, Adafruit va a seguir manteniendo las dos últimas versiones mayores.

Algunas de las tarjetas compatibles con CircuitPython, traen CircuitPython instalado. Otras, vienen preparadas para correr CircuitPython, pero este debe ser instalado. Así como también quieras actualizar la versión de CircuitPython que viene con la tarjeta. Los pasos para actualizar o instalar, son los mismos. Aquí vamos a cubrir como instalar y actualizar CircuitPython para tu tarjeta.

Solamente debes instalar CircuitPython UNA VEZ, y luego de esto puedes programar y editar tu código sin pasar por este proceso de nuevo, hasta que sea hora de actualizar.

¡Descarga la última versión!

Lo primero que deseas hacer, es descargar la versión más reciente de CircuitPython.

Si ya estás ejecutando CircuitPython, ¡verifica que estás ejecutando la última versión! Si no estás seguro, puedes seguir estos pasos para asegurarte que vas a tener la última versión instalada.

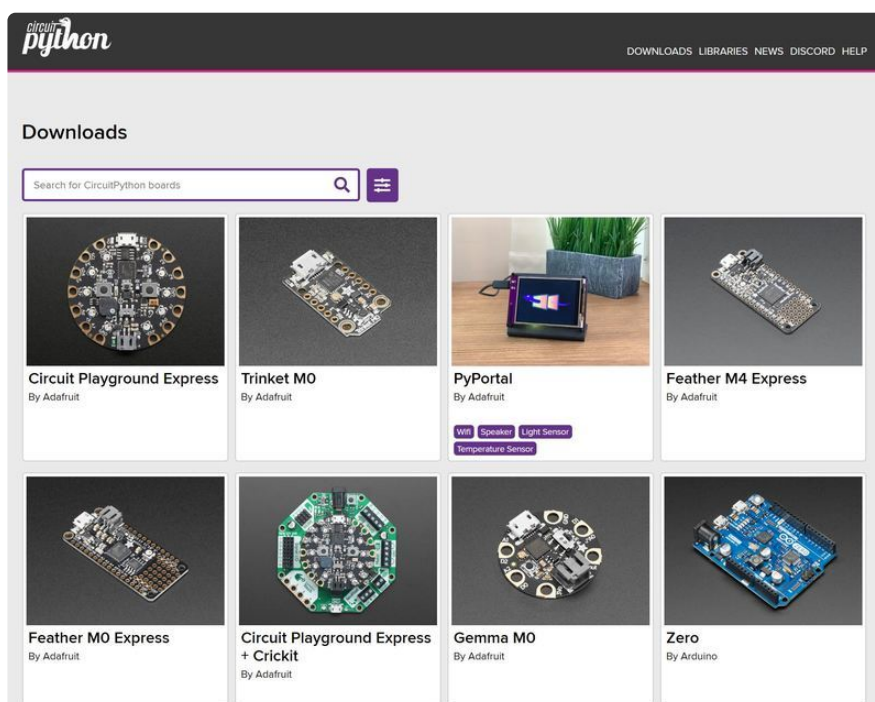
SIEMPRE RESPALDA TU CÓDIGO ANTES DE INSTALAR O ACTUALIZAR CIRCUITPYTHON. En la mayoría de los casos, nada va a ser eliminado de tu tarjeta

durante una actualización, Asegúrate de respaldarlo a tu computadora antes de seguir los pasos a continuación.

Descarga la última versión para tu tarjeta, dando click en el botón verde abajo, visitando [CircuitPython.org](https://adafru.it/Em8) (<https://adafru.it/Em8>).

Has click aquí para descargar
CircuitPython de CircuitPython.org

<https://adafru.it/Em8>



Luego, vas a querer conectar tu tarjeta con un cable USB verificado para transmitir datos. Existen algunos cables que funcionan solamente para carga, y pueden llevar a mucha frustración.

Controladores para Windows 7

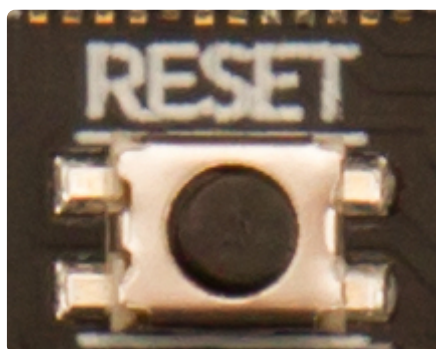
Si estás utilizando Windows 7, necesitas instalar un controlador antes de conectar la tarjeta.

Si estás utilizando Windows 7, utilice el enlace abajo, para descargar el paquete con el controlador. No es necesario instalar controladores en Mac, Linux o Windows 10.

Iniciando el gestor de arranque UF2

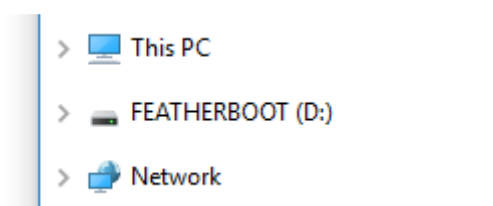
Casi todas las tarjetas para CircuitPython vienen con un gestor de arranque llamado UF2 (USB Flasher versión 2) que permite que el proceso de instalar o actualizar CircuitPython sea rápido y sencillo. El gestor de arranque es el modo que tu tarjeta necesita para que el archivo .uf2 de CircuitPython pueda ser cargado. Si el archivo que descargaste contiene el nombre de tu tarjeta en el nombre de archivo y termina en **uf2**, puedes continuar con esta sección. [Sin embargo, si el archivo termina en .bin, debes realizar un proceso de instalación más complejo; visita esta página para archivos tipo .bin \(\)](#).

Encuentra el botón de reset en tu tarjeta. Es un pequeño botón negro, y en la mayoría de las tarjetas, es el único botón disponible. (En la CircuitPython Express, es el botón más pequeño situado en el centro de la tarjeta).



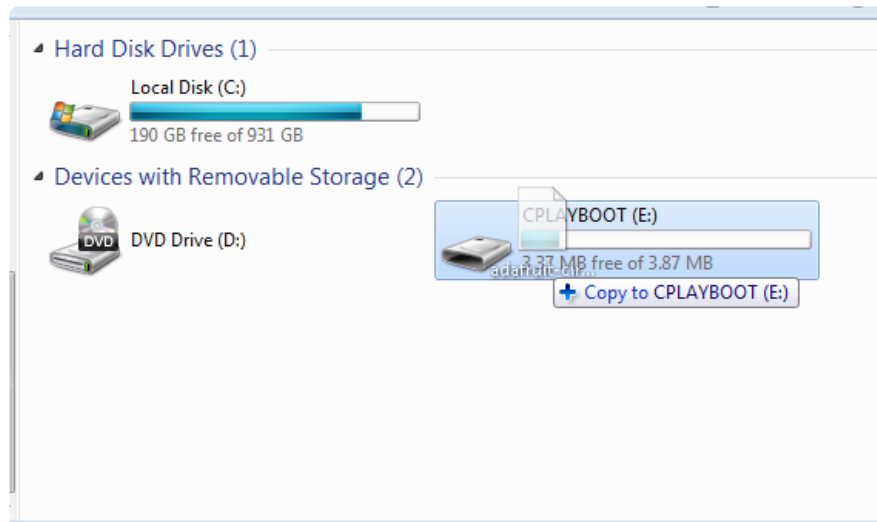
Presiona el botón dos veces para entrar en el gestor de arranque. Si no funciona al primer intento, no te desespere. El ritmo de estas presiones debe ser el correcto y alguna veces requiere algunos intentos. Si tienes una Circuit Playground Express recién salida de la bolsa, trata presionando el botón solo una vez.

Si funciona, el LED RGB en la tarjeta va a parpadear rojo y luego quedarse fijo en verde. La unidad de disco se va a llamar **nombretarjetaB00T**, donde **nombretarjeta** es el nombre definido para cada tarjeta. Por ejemplo una Feather va a tener **FEATHERB00T** y una Trinket va a tener **TRINKETB00T**, etc. En un futuro lo llamaremos solo **B00T**.

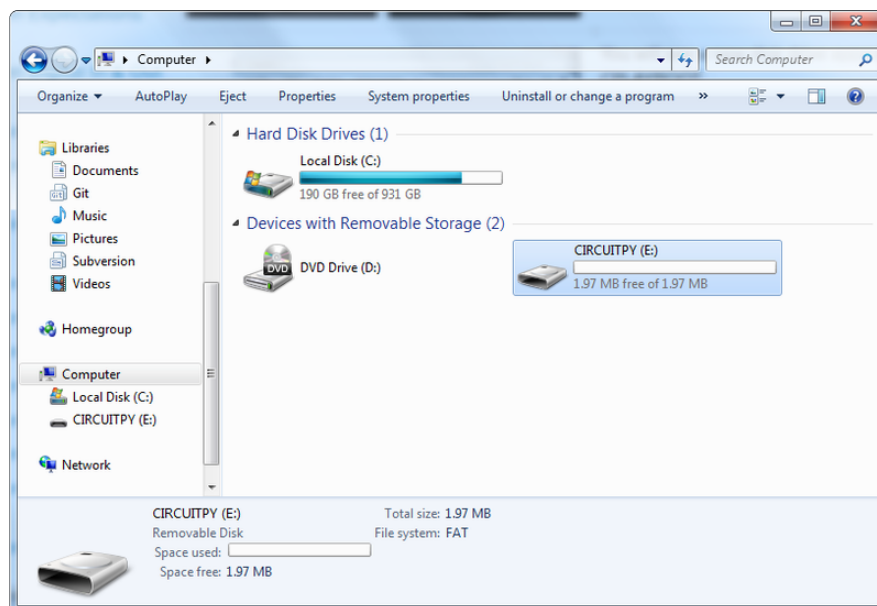


¡La tarjeta ya está en modo de gestor de arranque! Esto es lo que queremos para instalar CircuitPython.

Ahora encuentra el archivo que descargaste. Arrástralo hacia la unidad de disco del gestor UF2 llamada **B00T**.



Las luces van a parpadear de nuevo, la unidad **B00T** desaparece, y una nueva unidad de disco aparece en tu computadora con el nombre **CIRCUITPY**.



¡Felicidades! ¡Has instalado o actualizado CircuitPython de forma exitosa!

¿Cual es la diferencia entre **CIRCUITPY** y **nombretarjetaB00T**?

Cuando conectas una tarjeta CircuitPython a tu computadora, tu equipo va a ver la memoria flash de la tarjeta como una unidad de disco USB en el cual puedes almacenar archivo. Cuando instalas correctamente CircuitPython, verás la unidad de

disco **CIRCUITPY** . Cuando le das doble click al botón de reset, observarás a la unidad de disco **nombretarjetaBOOT** . Puedes arrastrarle archivos a ambos, pero solo **CIRCUITPY** va a correr tu código de CircuitPython.

Normalmente, cuando arrastras un archivo a una unidad de disco USB montada, el archivo se copia a la unidad y lo puedes observar con tu navegador de archivos. Sin embargo, cuando copia el archivo UF2 con CircuitPython a la unidad de disco **nombretarjetaBOOT** , parece que desaparece, y la unidad de disco se desconecta. ¡Esto es normal! El UF2 es básicamente un archivo instalador y no solo reside en la unidad, sino que instala CircuitPython si estamos en modo gestor de arranque (con **nombretarjetaBOOT**).

Usted va a poder copiar otros archivos a la unidad de disco de gestor de arranque (la **nombretarjetaBOOT**) pero no se van a ejecutar ni estarán disponibles en CircuitPython. ¡Así que asegúrate que una vez que completas la instalación de CircuitPython, que estás arrastrando y editando archivo de la unidad de disco **CIRCUITPY** !

Nombres de unidades de disco en modo gestor de arranque

Esta lista no es exhaustiva, pero te debería dar una idea de que buscar en el nombre de una unidad de disco en modo gestor de arranque.

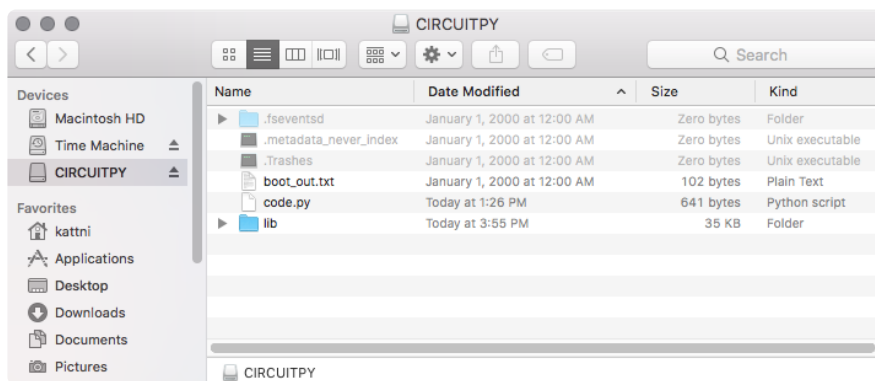
- Trinket M0 = **TRINKETBOOT**
- Gemma M0 = **GEMMABOOT**
- Circuit Playground Express = **CPLAYBOOT**
- ItsyBitsy M0 Express = **ITSYBOOT**
- ItsyBitsy M4 Express = **ITSYM4BOOT**
- Feather M0 Express = **FEATHERBOOT**
- Feather M4 Express = **FEATHERBOOT**
- Metro M0 Express = **METROBOOT**
- Metro M4 Express = **METROM4BOOT**
- Grand Central M4 Express = **GCM4BOOT**
- NeoTrelis M4 Express = **TRELM4BOOT**

La unidad de disco CIRCUITPY

Cuando CircuitPython termina de instalar o conectas una tarjeta CircuitPython a tu computadora con CircuitPython ya instalado, la tarjeta se muestra en tu computadora como un disco USB llamado **CIRCUITPY**.

La unidad de disco **CIRCUITPY** es donde tu código y librerías necesarias van a residir. Usted puede editar el código directamente en esta unidad de disco, y se va a ejecutar automáticamente. Cuando creas y editas código, usted salvará su código en el archivo llamado `code.py`, situado en la unidad de disco **CIRCUITPY**. Si estás siguiendo una guía del Learn, puedes pegar los contenidos de ejemplo del tutorial a este archivo `code.py` en tu unidad de disco **CIRCUITPY**, y lo salvas para ejecutar el ejemplo.

CircuitPython busca a `code.py` y ejecuta su contenido de forma automática cuando la tarjeta inicia, reinicia o cuando salvas el contenido del archivo. ¡Esto es lo que hace tan sencillo iniciar con proyectos y actualizar tu código!



Trabajando con múltiples dispositivos

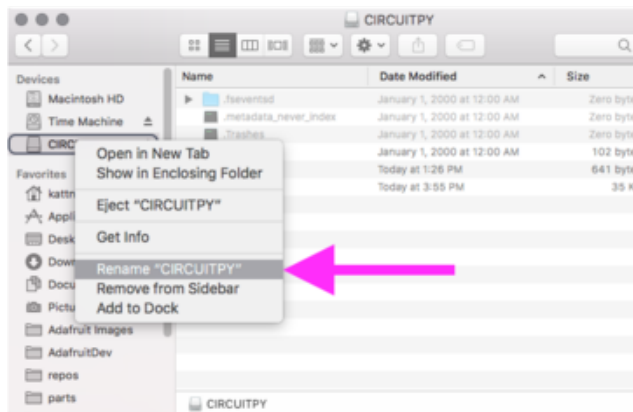
Existen muchas tarjetas que trabajan con CircuitPython. Usted se puede encontrar en una situación donde estás trabajando con más de una tarjeta a la vez. ¿Que sucede cuando conectas varias tarjetas a tu computadora? ¡Pues tienes múltiples unidades **CIRCUITPY**! ¿Ahora como sabes cual es cual? Usted puede renombrar cada unidad de disco **CIRCUITPY** para evitar confusión.

Renombrando CIRCUITPY

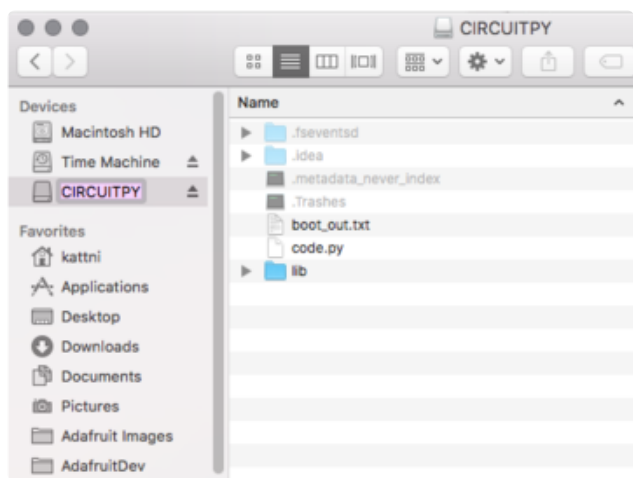
Cuando le cambias el nombre a **CIRCUITPY**, se escribe el nombre en el sistema de archivos. Esto significa que el cambio de nombre se va a mantener si desconectas la tarjetas. ¡Así como si recargas CircuitPython!

¡El nombre debe ser de 11 letras o menos! Esto es una limitante del sistema de archivo. Vas a recibir un error si escojes uno con más de 11 letras.

Renombrando CIRCUITPY en Mac



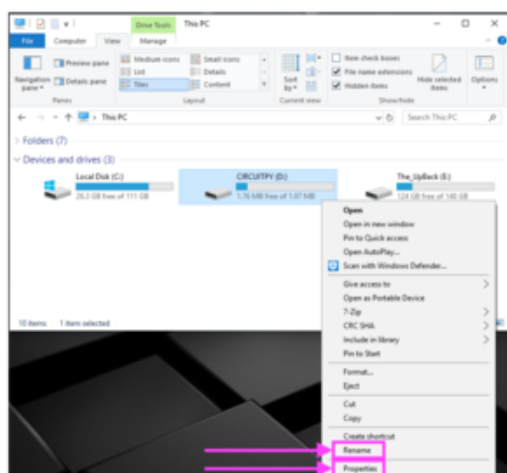
Cambiarle el nombre a tu unidad de disco **CIRCUITPY** es sencillo. Das click a tu unidad de disco en el Finder para ver su contenido. Luego le das click derecho a la unidad de disco en el Finder y escoges "Renombrar".



Cuando das click a "Renombrar", en el menú de click derecho, el nombre de la unidad de disco aparece en una caja de texto donde puedes renombrarla. Escribe el nuevo nombre.

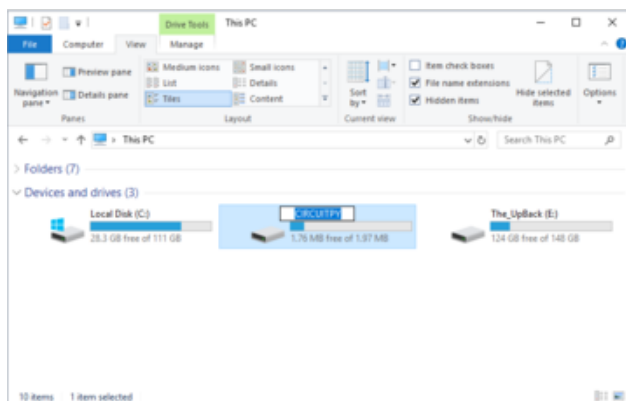
Renombrando CIRCUITPY en Windows

Cambiar el nombre a la unidad de disco **CIRCUITPY** en Windows es sencillo. Abre el Explorador de archivos y buscas la unidad de disco **CIRCUITPY**. Le das click derecho encima, y click a "Renombrar".



Renombrar la unidad de disco **CIRCUITPY** en Windows es sencillo. Abre el Explorador de archivos y buscas la unidad de disco **CIRCUITPY**. Le das click derecho encima, y click a "Renombrar".

También puedes renombrar la unidad de disco por medio del menú de "Propiedades" en el Explorador de Windows, incluyendo click derecho sobre la unidad de disco **CIRCUITPY** y escogiando "Propiedades".



Una vez que da click en "Renombrar" en el menú de click derecho, la unidad de disco aparece en un campo de texto donde puedes renombrarla. Escriba el nuevo nombre.

Renombrando CIRCUITPY en Linux

Para cambiar el nombre de la unidad de disco **CIRCUITPY** en Linux, es necesario ejecuta un par de pasos. Necesitas identificar el punto de monta, y luego ejecutar un comando para renombrar la unidad de disco.

Abra la aplicación de terminal. Ejecute el siguiente comando para consultar donde está montada tu tarjeta:

```
df | grep CIRCUITPY
```

Vas a ver **CIRCUITPY** en la derecha de la línea resultante. El punto de monta aparece a la izquierda de la línea y es similar a **/dev/foo** (donde **foo** es el nombre del punto de monta).

Ahora puedes ejecutar el siguiente comando para desmontar la tarjeta, reemplazando la palabra **foo** por tu punto de monta.

```
sudo umount /dev/foo
```

Para renombrar la tarjeta, ejecute lo siguiente:

```
sudo fatlabel /dev/foo NEW_NAME
```

Ahora desconecta tu tarjeta y la conectas para forzarla a remontarse con el nuevo nombre.

Para revisar que funcionó, busca el nuevo nombre de unidad de disco en tu navegador de archivos. O puedes ejecutar lo siguiente:

```
df | grep NEW_NAME
```

```
gandalf@work: ~  
gandalf@work:~$ df | grep CIRCUITPY  
/dev/sdb1 2024 1575 449 78% /media/gandalf/CIRCUITPY  
gandalf@work:~$ sudo umount /dev/sdb1  
gandalf@work:~$ sudo fatlabel /dev/sdb1 SHADOWFAX  
gandalf@work:~$  
gandalf@work:~$ df | grep SHADOWFAX  
/dev/sdb1 2024 1575 449 78% /media/gandalf/SHADOWFAX  
gandalf@work:~$
```

Renombrando CIRCUITPY desde CircuitPython

You can also rename the board using CircuitPython. Create a new file on your **CIRCUITPY** drive called `boot.py`. Copy the following code into the new `boot.py` file:

Usted también puede renombrar la unidad de disco **CIRCUITPY** utilizando CircuitPython. Crea un nuevo archivo en tu **CIRCUITPY** llamado `boot.py`. Copia el siguiente código dentro del archivo `boot.py`:

```
import storage  
  
storage.remount("/", readonly=True)  
  
m = storage.getmount("/")  
m.label = "NEW_NAME"  
  
storage.remount("/", readonly=False)
```

Desmonta tu tarjeta, y reiniciala ya sea presionando el botón de reset una vez, o desconectando la tarjeta y reconectándola. !Luego de un instante, debería aparecer en tu navegador de archivos con el `NUEVO_NOMBRE` que has escogido! Ya puedes borrar `boot.py` luego de que el nuevo nombre aparece en tu navegador de archivos.

Revirtiendo a CIRCUITPY

Puedes seguir el mismo proceso antes descrito para renombrar la unidad de disco, de nuevo hacia **CIRCUITPY**.

También puedes revertir al nombre **CIRCUITPY**, borrando el sistema de archivos. Si estás en una situación donde debe eliminar el sistema de archivos en tu tarjeta CircuitPython, la unidad de disco revertirá su nombre a **CIRCUITPY** cuando complete el borrado.

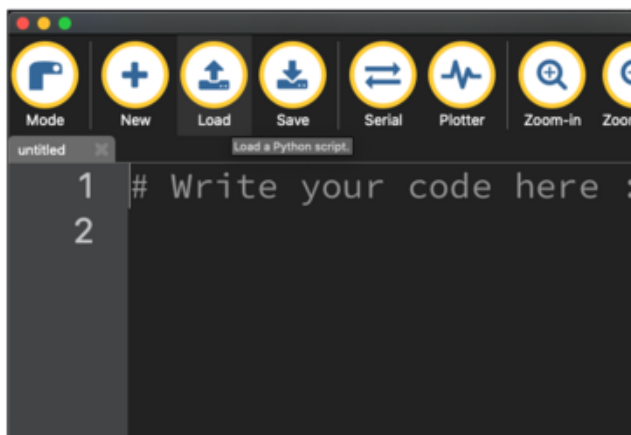
Creando y editando código

Una de las mejores cosas sobre CircuitPython, es lo sencillo que es poner a trabajar al código. En esta sección vamos a hablar de como crear y editar tu primer programa en CircuitPython.

Para crear y editar el código, todo lo que necesitas es un editor. Existen muchas opciones. **¡Nosotros recomendamos fuertemente a Mu! Es diseñado para CircuitPython, y es realmente simple y sencillo de utilizar, ¡con una consola serial integrada!**

Si no utilizas o no puedes usar Mu, hay editores básicos para texto en cualquier sistema operativo, como Notepad en Windows, TextEdit en Mac, y gedit en Linux. Sin embargo, muchos de estos editores no escriben sus cambios de inmediato a los archivos que editas. Esto puede causar problemas con CircuitPython. Ver la sección de [Editando código \(\)](#) a continuación. Si quieres saltarte dicha sección por el momento, asegurate de "Expulsar" o "Remove de forma segura" en Windows, o "sync" en Linux luego de escribir un archivo si no estás en Mu. (Esto no es un problema en MacOS).

Creando código



Abre tu editor y crea un archivo nuevo. Si estás usando Mu, da click al botón de Nuevo arriba a la izquierda

Copia y pega el siguiente código en tu editor:

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

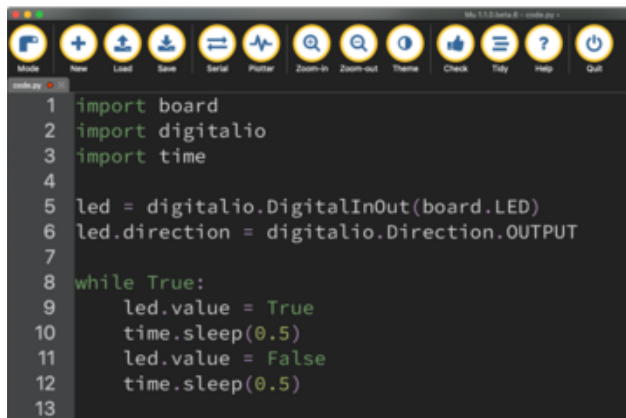
while True:
    led.value = True
    time.sleep(0.5)
```

```
led.value = False
time.sleep(0.5)
```

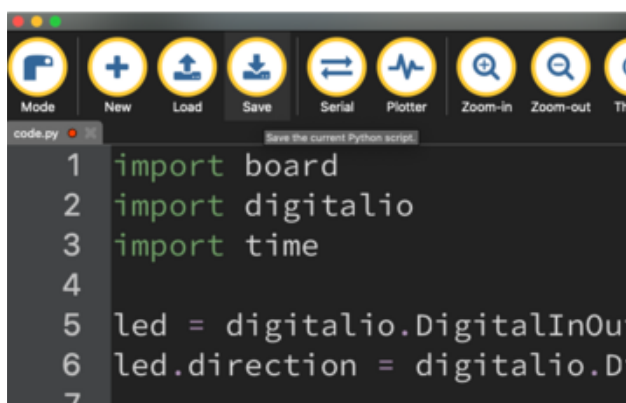
Si estás utilizando la CLUE de Adafruit, ¡vas a tener que editar el código para usar board.D17 como se muestra abajo!

Para las CLUE de Adafruit, vas a tener que utilizar `board.D17` en lugar de `board.D13`. El resto del código se mantiene igual. Realiza el siguiente cambio a la línea `led =`:

```
led = digitalio.DigitalInOut(board.D17)
```



Se va a ver Así - nota que cuando estás bajo la línea `while True:`, las siguientes cuatro líneas tienen espacios para indentarlas, pero están indentadas la misma cantidad. El resto de líneas no tiene espacios antes del texto.

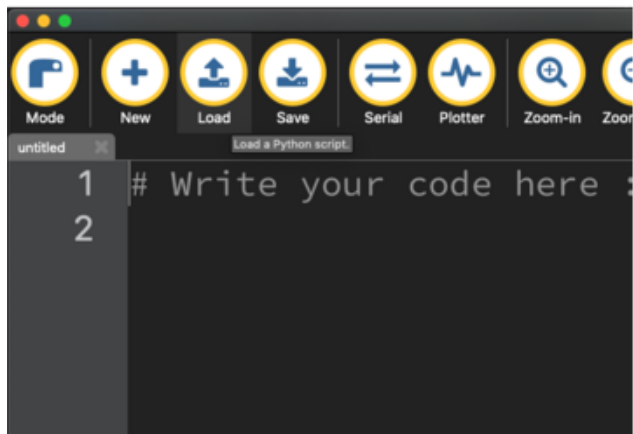


Salva este archivo como `code.py` bajo tu unidad de disco CIRCUITPY.

En cada tarjeta vas a encontrar un pequeño LED rojo. Debería estar parpadeando una vez por segundo.

¡Felicidades, estas ejecutando tu primer programa en CircuitPython!

Editando código



Para editar código, abre tu archivo **code.py** bajo tu unidad de disco **CIRCUITPY** usando un editor.

Realiza los cambios deseados en tu código. Salva el archivo. ¡Eso es todo!

Los cambios en tu código son aplicados en el momento que el archivo es salvado.

Solamente hay una advertencia para darte, antes de continuar...

¡No des click en Reset o desconectes!

El código CircuitPython en tu tarjeta detecta cuando los archivos cambian o son escritos y automáticamente re-inicia tu código. Esto hace programar muy rápido porque tu salvas y vuelve a correr.

¡Sin embargo, debes esperar a que el archivo esté completamente salvado antes de desconectar o reiniciar tu tarjeta! En Windows, usando algunos editores es posible que dure hasta 90 segundos o 30 segundos para Linux, para completar porque el editor de texto no salva el archivo completamente. Mac OS parece no tener este retraso, ¡lo cual es bueno!

Esto es muy importante para tener en cuenta. Si desconectas o reinicias la tarjeta antes de que tu computadora termine de escribir a la tarjeta, puedes corromper la unidad de disco. Si esto sucede, es posible que pierdas el código que has escrito, por lo que es importante respaldar tu código regularmente hacia una computadora.

Estas son algunas recomendaciones para evitar que suceda:

1. Utiliza un editor que escriba los archivos completamente cuando los salvas.

Editores recomendados:

- **mu** (<https://adafru.it/Be6>) es un editor que escribe de forma segura todos los cambios (Así como es también nuestro editor recomendado!)
- **emacs** (<https://adafru.it/xNA>) también es un editor que va a [escribir completamente los archivos cuando los salvas](#) ()
- **Sublime Text** (<https://adafru.it/xNB>) escribe de forma segura los cambios
- **Visual Studio Code** (<https://adafru.it/Be9>) parece que escribe de forma segura los cambios
- **gedit** en Linux, parece que escribe de forma segura los cambios
- **IDLE** (<https://adafru.it/IWB>), en Python 3.8.1 o posterior, [fue arreglado](#) (<https://adafru.it/IWD>) de forma que ahora escribe de forma segura los cambios

Recomendado solo para ciertos escenarios o con añadidos:

- **vim** (<https://adafru.it/ek9>) / **vi** escribe de forma segura los cambios. Pero configura a **vim** para que no escriba [archivos temporales](https://adafru.it/ELO) (<https://adafru.it/ELO>) (archivos .swp: registros temporales de las ediciones) para **CIRCUITPY**. Ejecuta vim con `vim -n`, configura la opción `no swapfile`, o configura la opción de directorio `(directory)` para que escriba los temporales en otro lugar. De otra manera, los archivos temporales van a causar reinicios de tu aplicación.
- El **IDE PyCharm** (<https://adafru.it/xNC>) escribe de forma segura si "Safe Write" se activa en Settings->System Settings->Synchronization (activado por omisión).
- Si estás utilizando **Atom** (<https://adafru.it/fMG>), instala el [paquete fsync-on-save](https://adafru.it/E9m) (<https://adafru.it/E9m>) para que siempre escriba los cambios en **CIRCUITPY**.
- **SlickEdit** (<https://adafru.it/DdP>) funciona solo si le [agregas un macro para que escriba los cambios a disco](#) (<https://adafru.it/ven>).

Nosotros no recomendamos estos editores:

- **notepad** (el editor por omisión en Windows) ni Notepad++ que pueden ser lentos para escribir, ¡por lo que recomendamos los editores mencionados arriba! Si estás utilizando notepad, Asegúrate de expulsar la unidad de disco (ver abajo).
- **IDLE** en Python 3.8.0 o anterior no fuerza la escritura inmediata de los cambio
- **nano** (on Linux) no fuerza la escritura de cambios
- **geany** (on Linux) no fuerza la escritura de cambios

- **Cualquier otro** - nosotros no hemos probado otros editores, Así que por favor ¡utilice uno recomendado!

2. Expulse o sincronice el dispositivo luego de escribir

Si estás utilizando uno de los editores no recomendados, ¡todavía hay esperanza! Todavía puedes lograr que funcione.

En Windows, puedes **Expulsar** o **Remover de forma segura**, la unidad de disco **CIRCUITPY**. No se va a expulsar realmente, pero forzará al sistema operativo a escribir tu archivo al disco. En Linux, puedes usar el comando **sync** en una terminal para forzar la escritura al disco.



!!!Oh No, hice algo mal y ahora la unidad de disco CIRCUITPY no aparece!!!

¡No se preocupe! El corromper una unidad de disco no es el fin del mundo (¡o de tu tarjeta!). Si esto sucede, sigue los pasos descritos en la página de [Solución de Problemas \(https://adafru.it/Den\)](https://adafru.it/Den) en la guía de cada tarjeta, para tenerla trabajando de nuevo.

Volviendo con la edición de código...

¡Ahora! Vamos a tratar de editar el programa que has agregado a tu tarjeta. Abre tu archivo **code.py** en tu editor. Vamos a realizar un simple cambio. Modifique el primer **0.5** hacia un **0.1** . El código debería verse de esta manera:

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(0.1)
    led.value = False
    time.sleep(0.5)
```

Mantenga el resto del código sin cambios. Salve su archivo. Observe que sucede con el LED de tarjeta. ¡Algo ha cambiado! ¿Sabes por qué? ¡Vamos a averiguarlo!

Explorando tu primer programa en CircuitPython

Primero, vamos a observar el código que estamos editando.

Aquí está de nuevo el código original:

```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

Imports y librerías

Cada programa en CircuitPython que ejecutas necesita tener mucha información para trabajar. La razón de porqué CircuitPython es tan sencillo es porque mucha de esa información se encuentra almacenada en otros archivos y trabaja detrás del telón. Estos archivos se llaman **librerías**. Algunas de ellas están integradas dentro de CircuitPython. Otras están almacenadas dentro de la unidad de disco **CIRCUITPY** en una carpeta llamada **lib**.

```
import board
import digitalio
import time
```

Las palabras de **import** le dicen a la tarjeta que vas a utilizar una librería en particular en tu código. En este ejemplo, hemos importado tres librerías: **board**, **digitalio**, y **time**. Estas tres librerías se encuentran integrada en CircuitPython, por lo que no son necesarios archivos adicionales. Eso es una de las cosas que lo vuelven un excelente primer ejemplo. ¡No necesitas nada extra para que funcione! **board** provee el acceso al hardware de tu tarjeta, **digitalio** permite acceder las entradas y salidas de hardware, **time** te permite pasar el rato 'durmiendo'

Configurando el LED

Las siguientes dos líneas de código configuran el código para utilizar el LED.

```
led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT
```

Tu tarjeta conoce el LED rojo como `D13` . Así que cuando inicializas este pin, lo preparamos para salida. Preparamos `led` para para que retenga esta información y no sea necesario escribirlo de nuevo más adelante en nuestro código.

Ciclos

La tercera sección comienza con una palabra de `while` . `while True:` básicamente significa "por siempre haga esto". `while True:` crea un ciclo. El código va a repetir "mientras" (while) la condición sea "verdadera" (true) (a diferencia de falso), y como `True` nunca es falso, el código se encicla sin fin. Todo el código que está indentado bajo `while True:` está "dentro" del ciclo.

Dentro del ciclo, tenemos cuatro elementos:

```
while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

Primero, tenemos `led.value = True` . Esta línea le dice al LED que se encienda. En la siguiente línea tenemos `time.sleep(0.5)` . Esta línea le dice a CircuitPython que realice una pausa durante medio segundo. Ya que esta línea está en medio del prendido y apagado el led, el led va a estar encendido por medio segundo.

Las siguientes dos líneas son similares. `led.value = False` le dice al LED que se apague, y `time.sleep(0.5)` le dice a CircuitPython que realice otra pausa, también medio segundo.

Luego el ciclo comienza de nuevo, y ¡se va a mantener ejecutándose siempre y cuando el código esté trabajando!

Entonces, cuando cambiaste el primer `0.5` hacia un `0.1` , bajaste el tiempo que el LED se mantiene encendido. Así que parpadea encendido, ¡muy rápido antes de apagarse!

¡Excelente trabajo! ¡Has editado código de un programa en CircuitPython!



¿Que pasa si no tienes el ciclo?

Si no tienes el ciclo, el código se va a ejecutar hasta el final, y terminal. Esto puede llevar a comportamientos extraños en programas sencillos como este, ya que el comportamiento de "exit" también reinicia el estado del hardware. Esto es un comportamiento diferente a ejecutar comandos en el REPL. Así que si estás editando un programa sencillo que parece

no funcionar, pueda que necesites agregar un ciclo al final para que el programa no termine.

El ciclo más sencillo sería:

```
while True:
```

```
    pass
```

Y recuerda, puedes presionar para salirte del ciclo.

Puedes también leer la sección [Comportamiento, en la documentación \(https://adafru.it/Bvz\)](https://adafru.it/Bvz).

Más Cambios

¡No hay razón para detenernos Aquí! Vamos a continuar. Cambie el segundo `0.5` por `0.1` para que se vea así:

```
while True:
    led.value = True
    time.sleep(0.1)
    led.value = False
    time.sleep(0.1)
```

¡Ahora parpadea muy rápido! ¡Has bajado ambos tiempos que el código prende y apaga el LED!

Ahora trata de cambiar ambos de `0.1` a `1`. El LED va parpadear mucho más despacio porque aumentaste la cantidad de tiempo que el LED esta prendido y apagado.

¡Bien hecho! ¡Lo estas haciendo muy bien! ¡Estas listo para comenzar con nuevos ejemplos y editarlos para ver que pasa! Estos cambios han sido sencillos, pero cambios mayores se puede realizar por el mismo proceso. Haces el cambio que deseas, lo salvas y recibes los resultados. ¡No tiene nada de complicado!

Nombrando tu programa

CircuitPython busca un código en tu tarjeta para ejecutar. Existen cuatro opciones: **code.txt**, **code.py**, **main.txt** y **main.py**. CircuitPython buscar por esos archivos, en ese orden, y ejecuta el primero que encuentre. Así como sugerimos utilizar **code.py** como tu archivo para código, es importante que sepas que existen otras opciones. Si pareciera que tu programa no se actualiza mientras trabajas, verifica que no hayas

creado otro archivo que pueda estar siendo leído en lugar del archivo sobre el que estas trabajando.

Conectándose a la Consola Serial

Algo de todos los días en CircuitPython (¡y en programación en general!) es algo llamado "print". Esto es una línea que incluyes en tu código para que tu código imprima texto. Un "print" en CircuitPython se ve así:

```
print("Hello, world!")
```

Esta línea resultaría en:

```
Hello, world!
```

Sin embargo, estos prints necesitan algún lugar para imprimirlos. ¡Aquí es donde entra la consola serial!

La consola serial recibe la salida de la tarjeta CircuitPython enviada por USB y la imprime para que la puedas ver. Esto es necesario cuando has incluido un print en tu código y deseas ver la salida. también es útil para mensajes de depuración de errores, porque tu tarjeta enviará errores y la consola serial los imprimirá.

La consola serial requiere una aplicación de terminal. Una terminal es un aplicación que provee una interfaz de texto para realizar ciertas tareas.

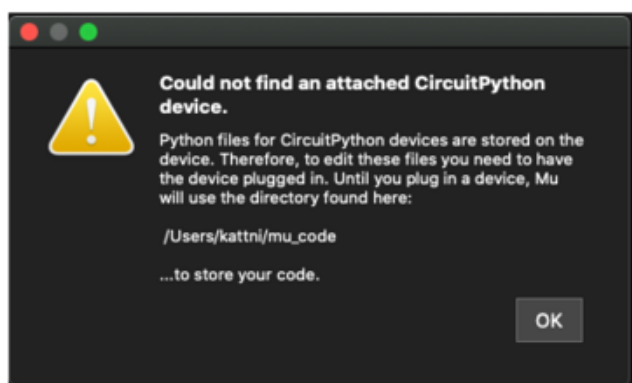
Si usas en Linux, y observas que la consola serial tarda varios segundos en conectarse, o si observas mensajes como "AT" o similares, entonces la aplicación modemmanager esta posiblemente interfiriendo. Elimínalo; no tiene mucha utilidad a menos que utilices modems de marcado telefónico. Para remover escribes este comando en el intérprete:

```
sudo apt purge modemmanager
```

¿Estás utilizando Mu?

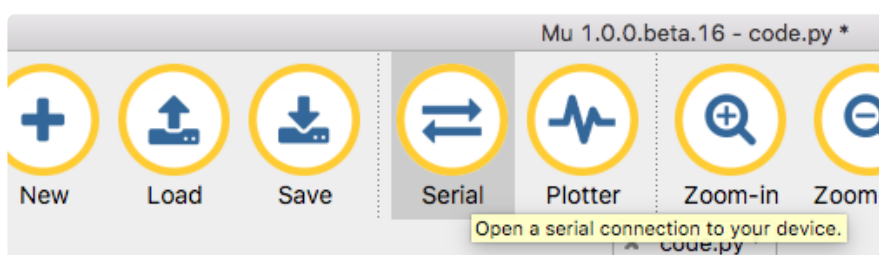
Si es así, ¡buenas noticias! La consola serial **está integrada en Mu** y va a **autodetectar tu tarjeta**, haciendo que usar el REPL sea realmente sencillo.

Favor notar que Mu todavía no trabaja con CircuitPython para tarjetas basadas en nRF52 o ESP8266, pasa a la siguiente sección para detalles de como utilizar una aplicación de terminal.



Primero, vamos a asegurarnos que tu tarjeta de CircuitPython está conectada. [Si estás en Windows 7, asegúrate de que has instalado los controladores. \(https://adafru.it/Amd\).](https://adafru.it/Amd)

Una vez en Mu, busque el botón **Serial** en el menú y dale click.



Configurando permisos en Linux

En Linux, si ves un mensaje de error similar a la que se encuentra abajo cuando aprestas el botón **Serial**, necesitas agrega a tu usuario al grupo que tiene permisos para conectarse con la consola serial.



En Ubuntu y Debian, te agregas al grupo **dialout** ejecutando:

```
sudo adduser $USER dialout
```

Luego de ejecutar el comando anterior, reinicia tu computadora para lograr acceso al grupo. En otras distribuciones de Linux, el grupo puede que tenga un nombre diferente. Puedes buscar detalles en la guía [Advanced Serial Console on Mac and Linux \(https://adafru.it/AAl\)](https://adafru.it/AAl), con información de como agregarte al grupo correcto.

¿Utilizando otra cosa?

Si por alguna razón no estás utilizando Mu para editar, estás utilizando una ESP8266 o nRF52 con CircuitPython, o si por alguna razón no te gusta la consola serial integrada, puedes ejecutar la consola serial como un programa aparte.

[Windows requiere que descargues una aplicación terminal, puedes buscar detalles en esta página \(https://adafru.it/AAH\).](https://adafru.it/AAH)

[Mac y Linux ambas tienen una integrada, aunque existe opciones disponibles para descargar, revisar esta página para más detalles \(https://adafru.it/AAl\)](https://adafru.it/AAl)

Interactuando con la Consola Serial

Una vez que te has conectado exitosamente a la consola serial, es hora de comenzar a utilizarla.

El código que escribimos anteriormente no tiene salida hacia la consola serial. Entonces, vamos a editarlo para crear alguna salida.

Abre el archivo **code.py** en tu editor, e incluye un **print**. ¡Puedes imprimir cualquier cosa que gustes! Solo incluye tu frase entre comillas, dentro de paréntesis. Por ejemplo:

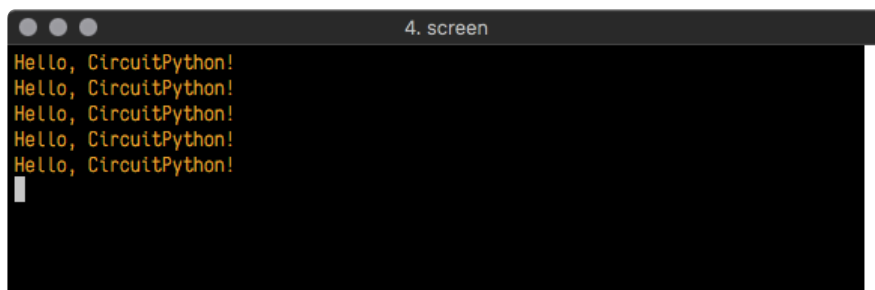
```
import board
import digitalio
import time

led = digitalio.DigitalInOut(board.D13)
led.direction = digitalio.Direction.OUTPUT

while True:
    print("Hello, CircuitPython!")
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(1)
```

Salva el archivo.

Ahora, vamos a mirar a la ventana con la conexión hacia la consola serial.



¡Excelente! ¡Nuestro print se está mostrando en la consola! Trata de cambiar el texto que imprime por otra cosa.

```
code.py
1 import board
2 import digitalio
3 import time
4
5 led = digitalio.DigitalInOut(board.D13)
6 led.direction = digitalio.Direction.OUTPUT
7
8 while True:
9     print("Hello back to you!")
10    led.value = True
11    time.sleep(1)
12    led.value = False
13    time.sleep(1)
14
```

Mantén la ventana con la consola serial donde la puedes observar. Salvar tu archivo. Vas a ver lo que la consola serial imprime cuando la tarjeta se reinicia. ¡Luego verás el nuevo cambio!

```
4. screen
Hello, CircuitPython!
Hello, CircuitPython!
Traceback (most recent call last):
  File "code.py", line 11, in <module>
KeyboardInterrupt:
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Hello back to you!
Hello back to you!
```

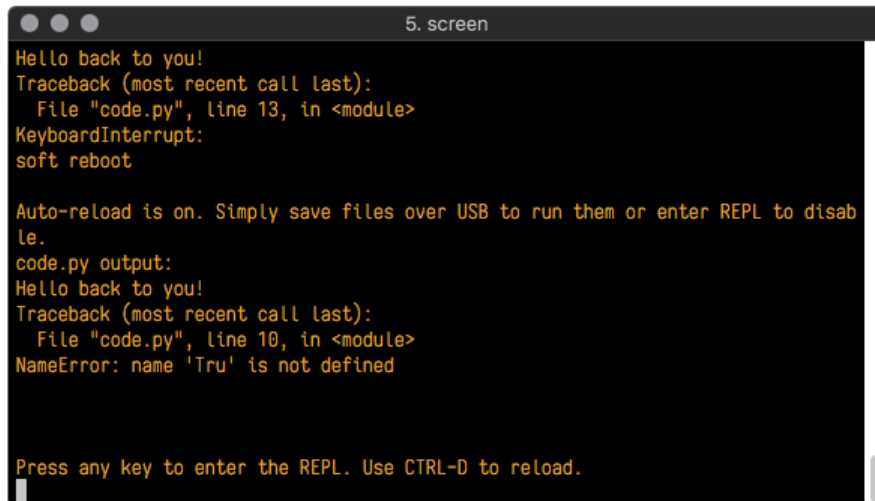
La línea que dice **The Traceback (most recent call last):** se refiere a la última cosa que tu tarjeta estaba realizando antes de salvar tu archivo. Esto es comportamiento normal y va a suceder cada vez que la tarjeta se reinicia. Esto es realmente útil para depuración. Vamos a introducir un error para ver como se utiliza.

Borra la **e** al final de **True** de la línea **led.value = True** de forma que diga **led.value = Tru**

```
code.py
1 import board
2 import digitalio
3 import time
4
5 led = digitalio.DigitalInOut(board.D13)
6 led.direction = digitalio.Direction.OUTPUT
7
8 while True:
9     print("Hello back to you!")
10    led.value = Tru
11    time.sleep(1)
12    led.value = False
13    time.sleep(1)
14
```


Salva el archivo. Vas a notar que el LED rojo ha dejado de parpadear, y puede que tengas un LED de estado de colores parpadeándote. Esto es porque el código ya no es correcto y no se puede ejecutar apropiadamente. ¡Necesitamos arreglarlo!

Usualmente cuando nos topamos errores, no es porque los hayas introducido a propósito. Puede que tengas 200 líneas de código y no tienes idea donde podría esconder tu error. Aquí es donde la consola sería nos puede ayudar. ¡Vamos a ver!



```
5. screen
Hello back to you!
Traceback (most recent call last):
  File "/code.py", line 13, in <module>
KeyboardInterrupt:
soft reboot

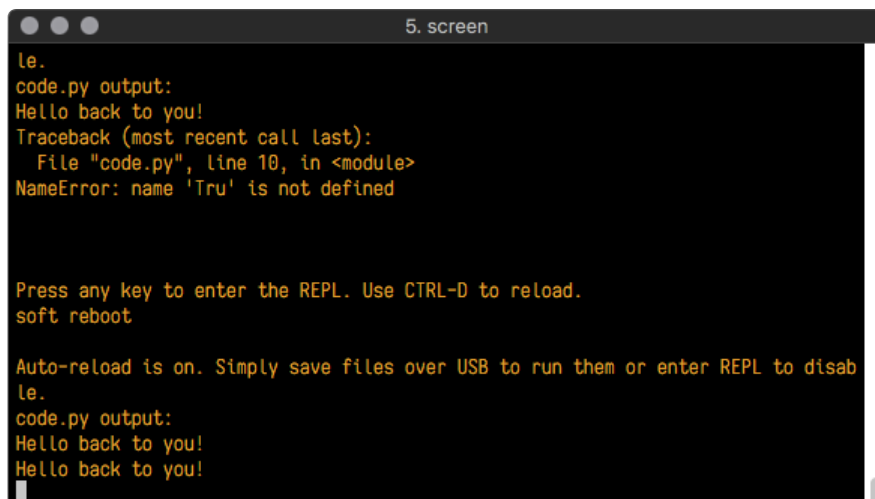
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Hello back to you!
Traceback (most recent call last):
  File "/code.py", line 10, in <module>
NameError: name 'Tru' is not defined

Press any key to enter the REPL. Use CTRL-D to reload.

```

La línea **The Traceback (most recent call last):** nos dice que es la última cosa que pudo ejecutar fue la línea 10 de tu código. El siguiente mensaje es tu error: **NameError: name 'Tru' is not defined**. Este error puede que no signifique mucho para ti, pero combinado con saber que el problema está en la línea 10, ¡te da un excelente lugar para comenzar!

Regresa a tu código, y observa la línea 10. Obviamente, ya sabes cual es el problema. Pero si no lo supieras, puedes tratar de mirar la línea 10 para ver si lo puedes encontrar. Si todavía no estás seguro, trata de buscar en internet el error para recibir ayuda. En este caso, sabes que buscar. Has escrito la palabra True mal. Arregla el error y salva el archivo.



```
5. screen
le.
code.py output:
Hello back to you!
Traceback (most recent call last):
  File "/code.py", line 10, in <module>
NameError: name 'Tru' is not defined

Press any key to enter the REPL. Use CTRL-D to reload.
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Hello back to you!
Hello back to you!

```

¡Buen trabajo arreglando el error! Tu consola serial está recibiendo información y tu LED rojo está parpadeando de nuevo.

La consola serial va a desplegar cualquier salida generada por tu código. Algunos sensores, como pueden ser sensores de humedad o thermo-resistores, reciben datos y puedes utilizar prints para desplegar dicha información. También puedes utilizar prints para depuración y buscar errores. Si tu código no funciona, y no sabes donde está fallando, puedes agregar prints en diferentes lugares para ver donde deja de imprimir.

¡La consola serial tiene muchos usos, y es una herramienta impresionante en general para aprendizaje y programación!

El REPL

Otra característica de la conexión serial es el **Read-Evaluate-Print-Loop** (Leer-Evaluar-Imprimir-Repite) o REPL. El REPL permite ingresar líneas individuales de código y ejecutarlas inmediatamente. Es muy útil para cuando estás teniendo problemas con un programar en particular y no puedes descifrar la razón. Es interactivo por lo cual es excelente para probar nuevas ideas.

Para usar el REPL, primero debes estar conectado en la consola serial. Una vez que la conexión está establecida, quieres presionar **Ctrl + C**.

Si había código ejecutándose, se detiene y vas a ver un mensaje

Press any key to enter the REPL. Use CTRL-D to reload. Sigue esas instrucciones, y presiona cualquier tecla en tu teclado.

El mensaje **The Traceback (most recent call last):** te dice la última cosa que tu tarjeta estaba haciendo antes de que tu apretaras Ctrl + C y la interrumpieras. La parte que dice **KeyboardInterrupt** es cuando presionaste Ctrl + C. Esta información puede ser útil para depuración, pero por ahora, no le vamos a poner atención. Solo ten en cuenta que es comportamiento normal.

```
2. screen
-0.306437 0.0 9.34634
-0.459656 0.0 9.49956
-0.459656 0.153219 9.49956
-0.306437 0.0 9.49956
-0.459656 0.0 9.34634
Traceback (most recent call last):
  File "code.py", line 24, in <module>
KeyboardInterrupt:

Press any key to enter the REPL. Use CTRL-D to reload.
```

Si no había código ejecutándose, vas a entrar en el REPL de forma inmediata luego de presionar Ctrl + C. No hay información de lo que estaba pasando anteriormente en tu tarjeta porque no había código ejecutándose.

```
2. screen
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.

Press any key to enter the REPL. Use CTRL-D to reload.
```

De todas formas, una vez que presiones una teclas vas a ver el mensaje de solicitud de `>>>` !dándote la bienvenida al REPL!

```
2. screen
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit CircuitPlayground Express with samd21g18
>>> 
```

Si tienes problemas obteniendo el mensaje de solicitud de `>>>`, trata de presionar Ctrl + C un par de veces más.

Lo primero que obtienes del REPL es información sobre tu tarjeta.

```
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit CircuitPlayground Express with samd21g18
```

Esta línea dice la versión de CircuitPython que estás utilizando y cuando fue liberada. Luego, te dice el tipo de tarjeta que estás utilizando y el tipo de microcontroladora que utiliza la tarjeta. Cada parte de esto puede que sea diferente para tu tarjeta dependiendo de las versiones que tengas.

Luego de esto, sigue el mensaje de solicitud de CircuitPython.

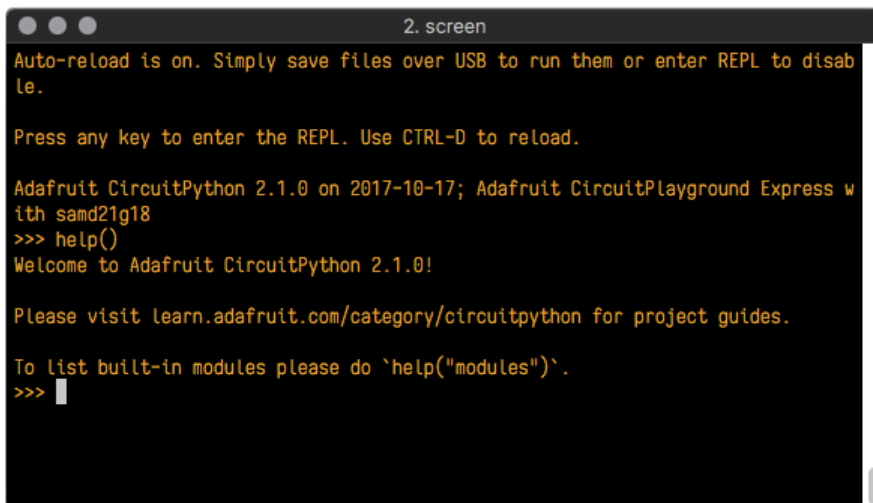
```
>>>
```

El mensaje de solicitud te está pidiendo todo tipo de comandos y código. Lo primero que vamos a hacer es ejecutar `help()`. Esto nos dice donde comenzar a explorar el REPL.

Para correr código en el REPL, escribes `help()` a la par del mensaje de solicitud del REPL.

```
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit Feather M0 Express with samd21g18
>>> help()
```

Ahora presionas enter y deberías ver un mensaje.



```
2. screen
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.

Press any key to enter the REPL. Use CTRL-D to reload.

Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit Circuit Playground Express with samd21g18
>>> help()
Welcome to Adafruit CircuitPython 2.1.0!

Please visit learn.adafruit.com/category/circuitpython for project guides.

To list built-in modules please do `help("modules")`.
>>>
```

La primera parte del mensaje es de nuevo una referencia a la versión de CircuitPython que estás usando. Segundo, el URL para guías de proyectos relacionados a CircuitPython. Luego... un momento... ¿Que es esto? `To list built-in modules, please do `help("modules")`` ¿Recuerdas las librerías que aprendiste mientras aprendías a crear código? ¡De esto es de lo que estaban hablando! Es un excelente lugar para comenzar, ¡vamos a ver!

Escribe `help("modules")` en el REPL y presiona enter.

```
3. screen
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit Feather M0 Express with sam
d21g18
>>> help()
Welcome to Adafruit CircuitPython 2.1.0!

Please visit learn.adafruit.com/category/circuitpython for project guides.

To list built-in modules please do `help("modules")`.
>>> help("modules")
__main__      busio          neopixel_write  time
analogio      digitalio      nvm              touchio
array         framebuffer    os               ucollections
audiobusio    gamepad        pulseio          ure
audioio       gc             random           usb_hid
bitbangio     math           samd             ustruct
board         microcontroller storage
builtins      micropython    sys
Plus any modules on the filesystem
>>>
```

Esto es una lista de las librerías núcleo integradas a CircuitPython. Ya hablamos como el módulo `board` contiene todos los pines en la tarjetas que puedes utilizar en tu código. Desde el REPL, ¡puedes ver esta lista!

Escribe `import board` en el REPL y presiona enter. Se irá a un nuevo prompt. Puede que parezca que nada ha sucedido, ¡pero ese no es el caso! Si recuerdas, la palabra `import` solo le dice al código que debe estar listo para realizar algo con ese módulo. En este caso, le está diciendo al REPL que planeas realizar algo con este módulo.

```
3. screen
d21g18
>>> help()
Welcome to Adafruit CircuitPython 2.1.0!

Please visit learn.adafruit.com/category/circuitpython for project guides.

To list built-in modules please do `help("modules")`.
>>> help("modules")
__main__      busio          neopixel_write  time
analogio      digitalio      nvm              touchio
array         framebuffer    os               ucollections
audiobusio    gamepad        pulseio          ure
audioio       gc             random           usb_hid
bitbangio     math           samd             ustruct
board         microcontroller storage
builtins      micropython    sys
Plus any modules on the filesystem
>>> import board
>>>
```

Ahora, escribe `dir(board)` en el REPL y presiona enter.

```
3. screen

Please visit learn.adafruit.com/category/circuitpython for project guides.

To list built-in modules please do `help("modules")`.
>>> help("modules")
__main__      busio          neopixel_write  time
analogio      digitalio      nvm             touchio
array         framebuffer    os             ucollections
audiobusio    gamepad        pulseio         ure
audioio       gc            random          usb_hid
bitbangio     math          samd            ustruct
board         microcontroller storage
builtins      micropython    sys
Plus any modules on the filesystem
>>> import board
>>> dir(board)
['A0', 'A1', 'A2', 'A3', 'A4', 'A5', 'SCK', 'MOSI', 'MISO', 'D0', 'RX', 'D1', 'TX',
 'SDA', 'SCL', 'D5', 'D6', 'D9', 'D10', 'D11', 'D12', 'D13', 'NEOPIXEL']
>>>
```

Esta es una lista de todos los pines que tienes disponible para utilizar en tu tarjeta. La lista va a ser diferente para cada tarjeta, dependiendo en la cantidad de pines disponibles. ¿Puedes encontrar **D13**? ¡Ese es el pin utilizado para parpadear el LED rojo!

El REPL también puede ser utilizado para ejecutar código. Pero ten en cuenta que **el código que digitas en él, no se salva en ningún lado**. Si estás probando algo nuevo que desear preservar, ¡asegúrate de salvarlo también en algún lado de tu computadora!

Todos los programadores en todos los lenguajes de programación comienzan realizando una porción de código que dice “¡Hola mundo!”. Vamos a decirle hola a otra cosa.

Escribe en el REPL:

```
print("Hello, CircuitPython!")
```

Y luego presiona enter.

```
>>> print("Hello, CircuitPython!")
Hello, CircuitPython!
>>>
```

¡Eso es todo lo necesario para ejecutar código en el REPL! ¡Buen trabajo!

Puedes escribir líneas individuales de código que corren solas. También puedes escribir programas completos en el REPL para probarlos. Como hemos dicho, sin embargo, recuerda que nada de lo que escribes en el REPL se salva.

Hay mucho que el REPL puede hacer por ti. Es excelente para probar nuevas ideas si quieres ver si unas cuantas líneas de código van a funcionar. Es fantástico para depuración de código, mientras escribes las líneas una por una para encontrar donde falla. Te permite ver cuales librerías hay disponibles y explorar estas librerías.

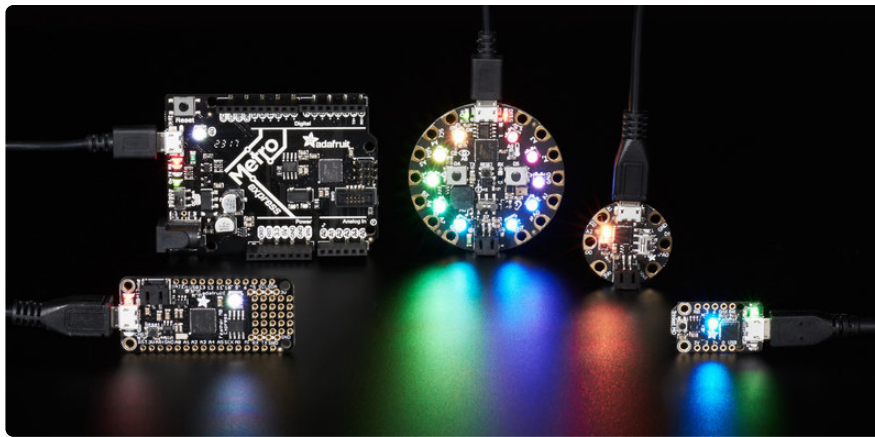
¡Trata de escribir más en el REPL a ver que pasa!

Retornando a la consola serial

Cuando estás listo para abandonar el REPL y regresar a la consola serial, simplemente presionas **Ctrl + D**. Esto va a recargar tu tarjeta y re-activar la consola serial. Esto va a reiniciar el programa que tenías ejecutando antes de entrar en el REPL. En la ventana de la consola, vas a ver la salida del programa que tenías ejecutando. Y si tu programa estaba afectando algo visible en la tarjeta, también observarás donde inicia.

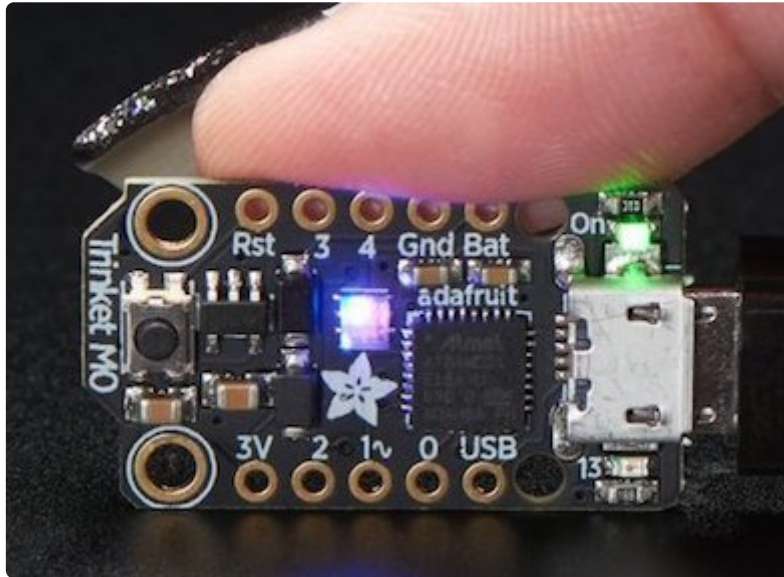
¡Puedes regresar al REPL en cualquier momento!

Hardware para CircuitPython



¡Ahora es tiempo de hacer algo grande con lo que has aprendido! Toda tarjeta de CircuitPython es perfecta para proyectos. Sin embargo, cada una brilla en diferentes áreas. Te vamos a dar algunos detalles sobre cada tarjeta, y traer a luz Guías de Aprendizaje donde cada una ha sido utilizada. ¡Usted puede probarlas u obtener ideas para su proyecto!

Trinket M0

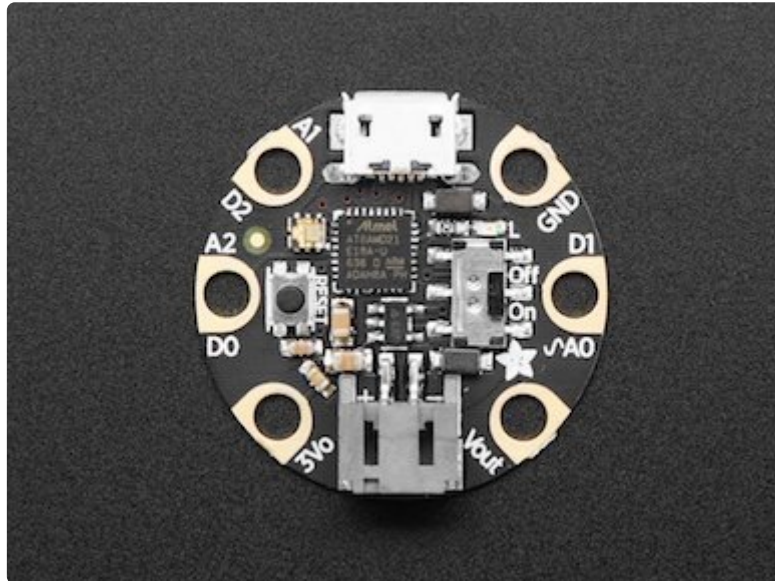


El [Trinket M0 de Adafruit \(http://adafru.it/3500\)](http://adafru.it/3500) es la tarjeta para CircuitPython más grande que tenemos. ¡Pero que no te engañe! Es una tarjeta pequeña con mucho poder. Queríamos diseñar una tarjeta de un tamaño pequeño para que quepa en cualquier proyecto, de bajo costo para usar sin duda alguna. ¿Planea realizar una prueba de concepto y necesita una tarjeta con CircuitPython para agregarle? ¿No estás listo para desarmar el proyecto en el que trabajaste tanto para entraerle la tarjeta que usaste la última vez? La Trinket M0 es para esto. Es la tarjeta disponible de menor costo para CircuitPython, ¡pero es un buen contrincante a las tarjetas más grandes!

Las Trinket M0 viene con CircuitPython instalado y con código de demostración en la tarjeta. Puedes abrir y editar el archivo `main.py` que encuentras en la unidad de disco CircuitPython, ¡para iniciar o crear tu propio! La [guía para la Trinket M0 \(\)](#) te brinda todo lo que necesitas saber sobre tu tarjeta. Revisa los [ejemplos en la sección de CircuitPython \(\)](#) para encontrar una enorme lista de ejemplos a probar.

Puedes utilizar una Trinket M0 para construir un robot para enfriar bebidas llamado [“Chilled Drinkibot” \(\)](#) que utiliza la Trinket para controlar un dispositivo de termoeléctrico para enfriar una bebida. O un tenebroso proyecto de Halloween que convierte un contenedor para confites, [¡en un caldero que grita! \(\)](#)

Gemma M0

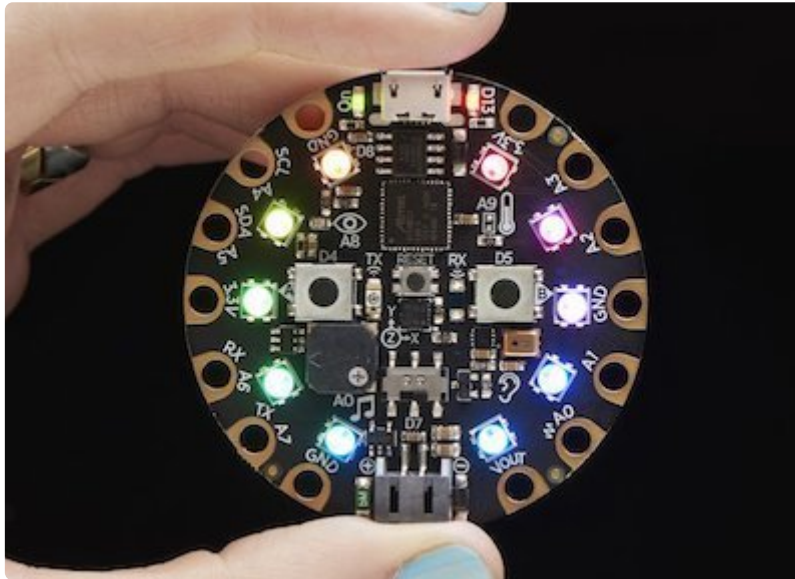


La [Gemma M0 de Adafruit \(\)](#) es una pequeña tarjeta para CircuitPython con apenas suficientes cosas integradas para construir muchos proyectos simples. Está diseñada para trabajar con proyectos electrovestibiles, con grandes huecos alrededor del exterior para poder coser en ellos (¡también con amigables con clips estilo lagarto!). Las Gemma M0 van a subir de nivel tus electrovestibiles, siendo más sencillas de usar que nunca. Tiene superficies capacitivas que detectan cuando las tocas, un switch on-off, y un LED RGB DotStar integrado en la tarjeta para que tengas mucho que hacer sin necesidad de agregarle nada. ¡Si le agregas hilo conductivo y LEDs vas a tener un electrovestible parpadeando en poco tiempo!

Similar a las Trinket, la Gemma M0 viene con CircuitPython y tiene código de ejemplo en la tarjeta. ¡Puedes abrir y editar el archivo main.py en la unidad de disco **CIRCUITPY**, o crear el tuyo! La [guía de la Gemma M0 \(\)](#) te muestra toda la información sobre la tarjeta, y tiene una excelente [lista de ejemplos con CircuitPython \(\)](#) para probar.

Utilice la Gemma M0 para crear un par de los llamados [Clockwork Goggles \(\)](#) con divertidos patrones de luz en anillos NeoPixel. ¡O puedes crear accesorios para ropa como este pendiente impreso llamado [Pendiente Sheikah \(\)](#) para agregar un poco de luz a tu siguiente disfraz!

Circuit Playground Express

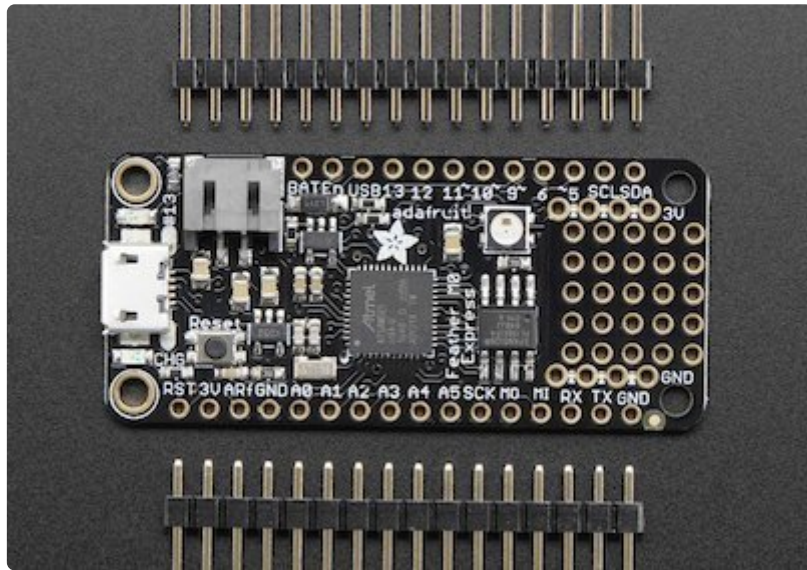


La [Circuit Playground Express de Adafruit \(\)](#) es el siguiente paso hacia una perfecta introducción a electrónica y programación. Viene repleta de sensores, LEDs, botones y switches, ¡y es super fácil de comenzar a usar! Esta tarjeta es muy versátil. Sin importar si eres nuevo a electrónica y programación, o un veterano, la Circuit Playground Express es una impresionante tarjeta para trabajar. Con tanto integrado en la tarjeta, puedes aprender como trabajan diferentes tipos de componentes electrónicos funcionan y aprender como programarlos sin necesidad de comprar otros componentes. ¡Solo necesitas un cable USB y a la tarjeta! Pero, eso es solo el principio. Muchas de las superficies alrededor de la tarjeta funcionan de muchas formas, permitiéndote conectar otras cosas a la tarjeta. Por ejemplo puedes conectar un motor tipo servo o un potenciómetro. ¡Las posibilidades son infinitas!

La [guía de Circuit Playground Express \(\)](#) contiene toneladas de información sobre todas las características impresionantes de la tarjeta. La [sección de CircuitPython \(\)](#) de la guía tiene una amplia lista de ejemplos utilizando las características integradas de CircuitPython y de la tarjeta. Hay una sección llamada [Python Playground \(\)](#) con más demostraciones y un proyecto de Caja de Ritmos o [Drum Machine \(\)](#) para probar.

Puedes convertir tu Circuit Playground Express usando el toque capacitivo en un [Piano en Llave de Lima \(\)](#) utilizando las superficies táctiles de la tarjeta. ¡Utilizando el acelerómetro integrado puedes crear un Platillo Volador OVNI (<https://adafru.it/BeJ>) completo con luces y sonidos alienígenas utilizando tu tarjeta, y algunos materiales que encuentras en la casa o con un platillo volador impreso 3d!

Feather M0 Express

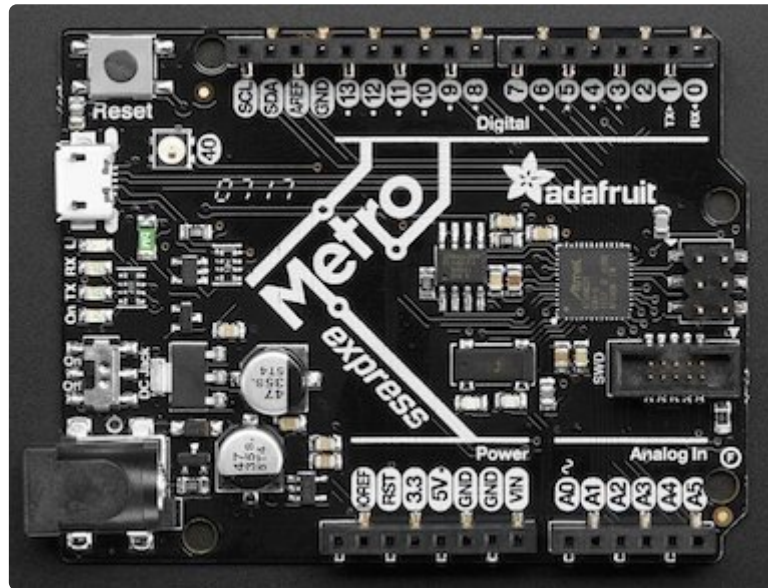


La [Feather M0 Express de Adafruit \(\)](https://adafruit.com/product/4077) es la primera Feather diseñada específicamente para CircuitPython. Es parte de la línea de tarjetas Feather de Adafruit para desarrollo (<https://adafru.it/BeK>) diseñadas para trabajar solas o en pila, y con alimentación por USB o baterías de iones de litio, así que funciona tanto para proyectos estacionarios o portátiles. La Feather M0 Express viene con dos cabeceras para pines para utilizar placas de pruebas, o puedes soldarle cables directamente a los pines de la tarjeta. Esto permite prototipado mientras trabajas en tu proyecto y una solución más permanente para cuando te sientas a gusto. Una de las cosas que hace a las Feather M0 Express impresionante es la cantidad de tipo de tarjetas llamadas [FeatherWings \(\)](#) las cuales están diseñadas para encajar en las Feather. Existen librerías para CircuitPython para muchas de las tarjetas y más se desarrollan todo el tiempo.

Las Feather M0 Express vienen listas para CircuitPython con el gestor de arranque UF2 instalado a la espera de que le instales CircuitPython cuando recibes la tarjeta. ¡Crea tu primer programa, sávalo a la tarjeta y listo! La guía de Feather M0 Express (<https://adafru.it/BeM>) tiene todos los detalles sobre tu tarjeta, y tiene una sección de CircuitPython (<https://adafru.it/BeN>) que te ayudará a iniciar.

Las Feather M0 Express se pueden utilizar para todo tipo de proyectos. Construye un [Píxel LED POV con CircuitPython \(\)](#) utilizando piezas impresas 3D y tiras DotStar. Puedes crear un Letrero Acrílico LED (<https://adafru.it/BeP>) con iluminación lateral a un grabado utilizando NeoPixels. Hay guías que acompañan a las Featherwings y explican como utilizarlas con CircuitPython como la [Pantalla OLED \(\)](#) y la [Featherwing Adalogger \(\)](#).

Metro M0 Express



La [Metro M0 Express \(\)](#) es la primera tarjeta Metro diseñada para trabajar con CircuitPython. Esta no es una tarjeta para principiante. Si apenas estás comenzando, te recomendamos alguna de las tarjetas anteriores. Tiene muchas de las mismas características de la Feather M0 Express, así como capacidades específicas para desarrollo (¡Como un puerto SWD integrado!). La Metro M0 Express se diseñó para tener un formato físico compatible con Arduino, por lo que si tienes shields para Arduino, esta tarjeta será genial para tí. Ya existen librerías para CircuitPython para muchos de los shields. Tiene 25 pines GPIO (¡la mayoría de muchas de estas tarjetas!) así que es excelente para alguien que está buscando muchas opciones.

Las Metro M0 Express viene lista para CircuitPython con el gestor de arranque UF2 instalado y está lista para que le instales CircuitPython cuando recibas la tarjeta. ¡Crea tu primer programa, lo salvas a la tarjeta y listo! La guía sobre la Metro M0 Express (<https://adafru.it/BeS>) te brinda detalles sobre tu tarjeta, y la [sección de CircuitPython \(\)](#) está disponible para ayudarte a iniciar.

Todos los sensores y tarjetas breakout con librerías para CircuitPython van a funcionar con la Metro M0 Express corriendo CircuitPython. Encuentre la guía para tu sensor y la sigues para encontrar como cablearlo. Hay muchas opciones disponibles.

¿Que Sigue?

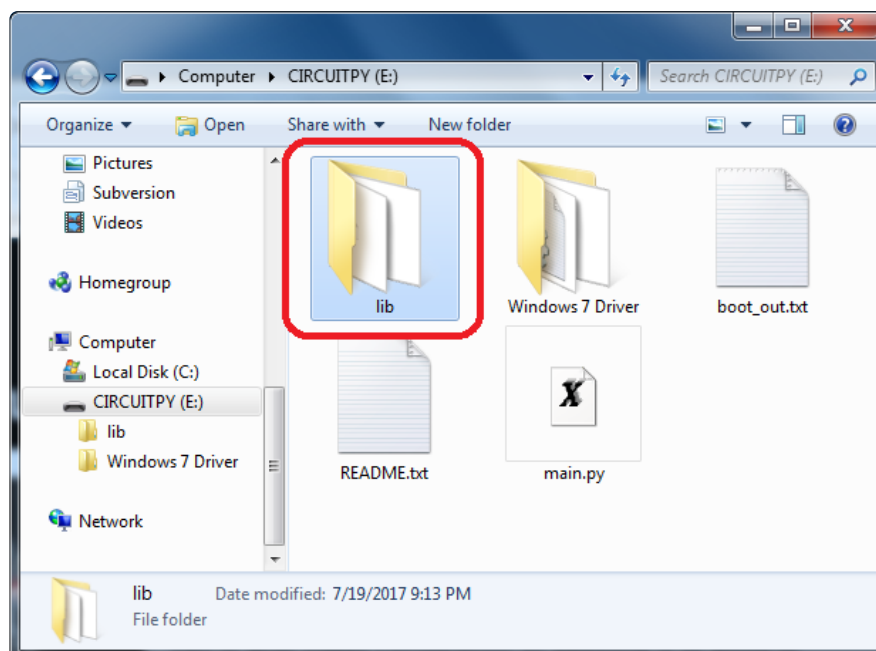
Ya estás listo para adentrarte en más Guías de Aprendizaje o simplemente iniciar con un nuevo proyecto. ¡Buen trabajo y buena suerte!

Librerías para CircuitPython

Mientras continuamos desarrollando CircuitPython y creando nuevas versiones, vamos a ir dejando de dar mantenimiento a las versiones más viejas. Visite <https://circuitpython.org/downloads> para descargar la última versión de CircuitPython para tu tarjeta. Debes descargar las Librerías Agrupadas para CircuitPython correctas para tu versión de CircuitPython. Favor actualice CircuitPython y luego visite <https://circuitpython.org/libraries> para descargar la última versión de las Librerías Agrupadas.

Cada programa en CircuitPython que ejecutas necesita tener mucha información para operar. La razón por la que CircuitPython es tan sencillo de usar es que la mayoría de la información reside en otros archivos y trabaja en el fondo. Estos archivos se llaman librerías. Algunas están integradas en CircuitPython. Otras están almacenadas en la unidad de disco **CIRCUITPY** en una carpeta llamada **lib**. Parte de lo que hace CircuitPython tan asombroso es la habilidad de almacenar código de forma separada al firmware. El almacenar código separado al firmware hace que sea más fácil actualizar tanto el código que escribes como las librerías de las que dependes.

Tu tarjeta puede que ya venga con la carpeta **lib**, en la base de la unidad de disco. Sino, simplemente crea la carpeta. Cuando instalas CircuitPython por primera vez, una carpeta vacía **lib** va a ser creada para ti.



Las librerías de CircuitPython trabajan de la misma forma que los módulos de Python regular, así que la [documentación de Python \(\)](#) es una excelente referencia sobre como funciona todo. En términos de Python, podemos poner nuestros archivos de librería en la carpeta **lib** porque es parte del path por omisión de Python.

Una desventaja de esta forma de manejar las librerías es que no vienen incluidas. Para utilizarlas uno debe copiarlas a la unidad de disco **CIRCUITPY** antes de que puedan ser usadas. Por fortuna, nosotros proveemos un agrupado de librerías.

Nuestro agrupado y sus versiones también son versiones optimizadas de las librerías con la extensión de archivo **.mpy**. Estos archivos requieren menos espacio en la unidad de disco y requieren menos memoria al cargarse.

Instalando el Agrupado de Librerías de CircuitPython

Nosotros estamos constantemente actualizando y mejorando nuestras librerías, así que no (por ahora) les ponemos el empaquetado de librerías a las tarjetas cuando las enviamos. En cambio, puedes encontrar código ejemplo en las guías para tu tarjeta el cual depende de librerías externas. Algunas de estas librerías van ser ofrecidas por nosotros en Adafruit, ¡y puede que otras sean escritas por miembros de la comunidad!

De cualquier manera, mientras comienzas a explorar CircuitPython, vas a querer aprender como cargarle librerías a la tarjeta.

Ahora descarga la última versión del Agrupado de Librerías de Adafruit para CircuitPython dándole click al botón abajo.

Nota: La versión del agrupado debe ser la versión correcta para la versión de CircuitPython que ejecutas – usas la el agrupado versión 3.x para CircuitPython 3, y usar el agrupado versión 4.x para CircuitPython 4, etc. Si mezclas versiones de librerías y versiones mayores de CircuitPython, lo más probable es que recibas errores debido a cambio en las interfaces hacia las librerías durante los cambios de versión.

Has Click para la última versión del
Agrupado de Librerías de Adafruit
para CircuitPython

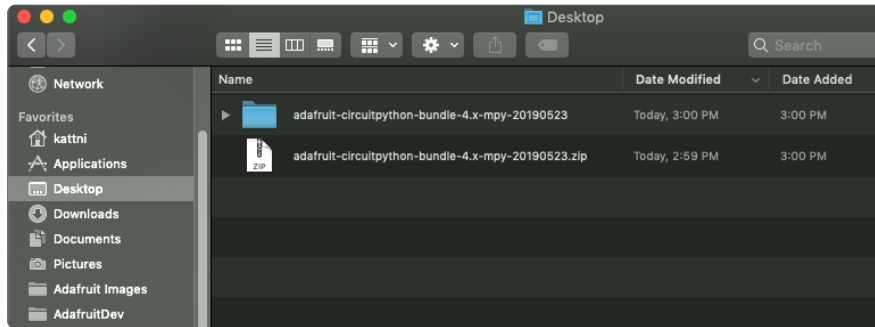
<https://adafru.it/ENC>

Si necesitas otra versión, puedes visitar la [página de versiones agrupadas](#) () que te va a permitir seleccionar exactamente la versión que andas buscando, así como contiene información sobre cambios.

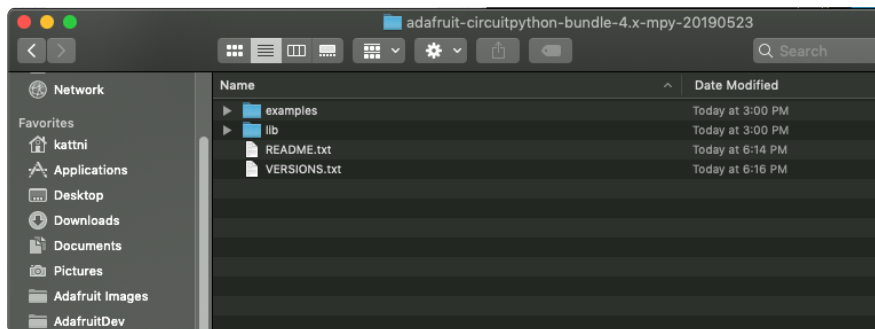
De cualquier forma, descarga la versión correcta para la versión de firmware de **CircuitPython que usas**. Si no conoces la versión, mira la información inicial en el REPL de CircuitPython, el cual reporta la versión. Por ejemplo, si estás ejecutando v4.0.1, descarga la versión 4.x del agrupado de librerías. También existe un agrupado

de **py**, el cual contiene los archivos de python sin compresión, lo cuales probablemente no quieres a menos que estés realizando un trabajo avanzado sobre las librerías.

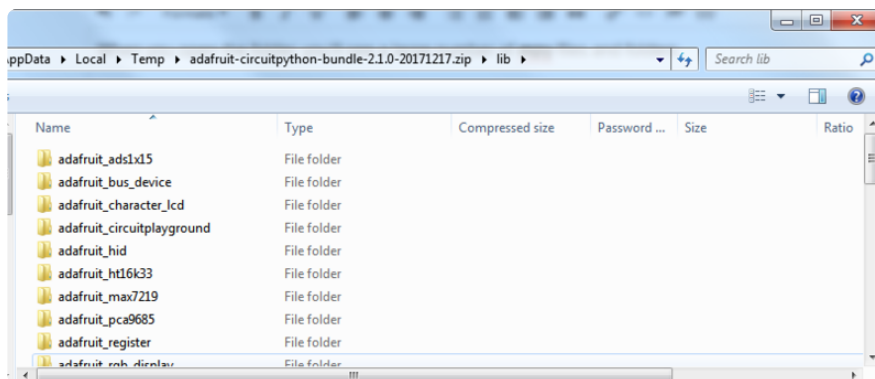
Luego de descargar el archivo zip, extrae sus contenidos. Esto se hace usualmente dándole doble click al archivo zip. En Mac OSX, el archivo termina en la misma carpeta que el archivo zip.



Abre la carpeta del empaquetado. Dentro vas a encontrar dos archivos informativos, dos carpetas. Una es la carpeta lib con el agrupado, y la otra es una carpeta con ejemplos para las librerías del agrupado.



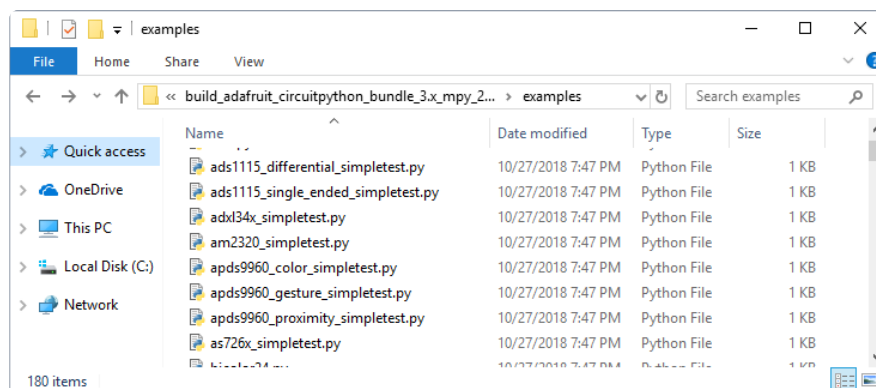
Ahora abre la carpeta lib. Cuando abres la carpeta, vas a ver una gran cantidad de archivos **mpy** y carpetas.



Archivos Ejemplo

Todos los archivos de ejemplo de cada librería ahora se incluyen en el agrupado, así como también existe un agrupado con solo ejemplos. Estos se incluyen por dos razones:

- Permiten pruebas rápidas sobre dispositivos.
- Proveen un ejemplo para basar el código, el cual se construye fácilmente sobre la base hacia propósitos particulares.



Copiando librerías a tu tarjeta

Primero vas a querer crear la carpeta **lib** en la unidad de disco **CIRCUITPY**. Abre la unidad de disco, das click derecho y buscas la opción para crear una carpeta, nombrándola **lib**. Ahora, abre la carpeta **lib** que has extraído del archivo zip descargado. Dentro vas a encontrar una cantidad de carpetas y archivos **.mpy**. Encuentra la librería que deseas utilizar y realiza la copia a la carpeta **lib** de la unidad de disco **CIRCUITPY**.

Esto también aplica para los archivos de ejemplo. Se proveen solo como archivos crudos **.py**, por lo cual es posible que sea necesario convertirlos a **.mpy** usando la herramienta **mpy-cross**, si encuentras errores de **MemoryErrors**. Esto ha sido contemplado en la [Guía de Esenciales para CircuitPython \(\)](#). El uso es el mismo descrito arriba en la sección de las tarjetas Express. Nota: Si no colocas los ejemplos en una carpeta separada, deberías remover los ejemplos de los **import**.

Ejemplo: **ImportError** debido a librerías faltantes

Si decides descargar las librerías mientras las necesitas, es posible que quieras escribir código que trate de utilizar una librería que aún no has cargado. Vamos a demostrar que sucede cuando tratas de utilizar una librería que no has carga a tu tarjeta, y revisar los pasos necesarios para resolver el problema

Esta demostración solo va a devolver un error si no tienes la librería requerida instalada en la carpeta **lib** de la unidad de disco **CIRCUITPY**.

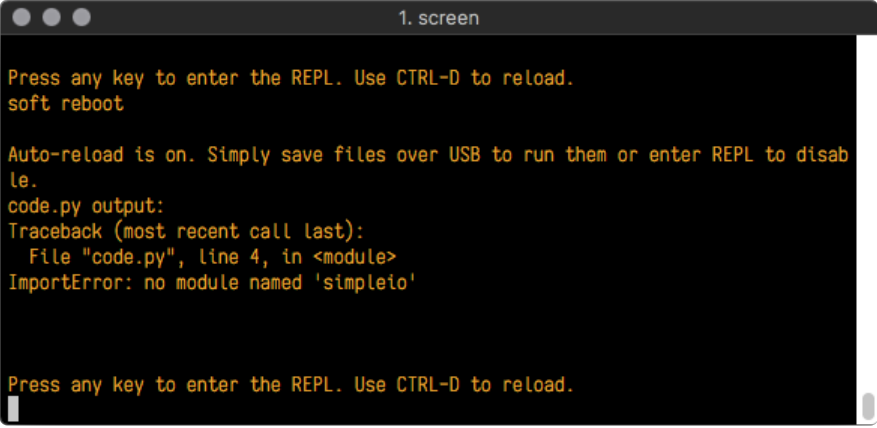
Vamos a modificar la versión del ejemplo que parpadea.

```
import board
import time
import simpleio

led = simpleio.DigitalOut(board.D13)

while True:
    led.value = True
    time.sleep(0.5)
    led.value = False
    time.sleep(0.5)
```

Salve este archivo. Nada sucede con la tarjeta. Vamos a revisar la consola serial para ver que está pasando.



```
1. screen

Press any key to enter the REPL. Use CTRL-D to reload.
soft reboot

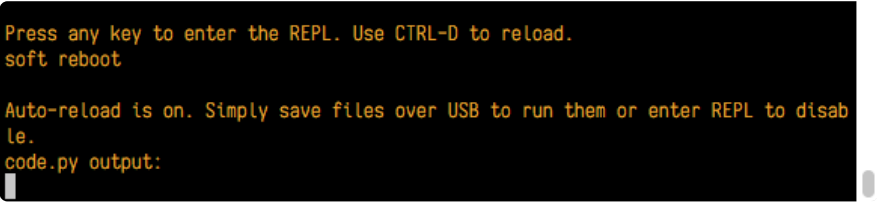
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
Traceback (most recent call last):
  File "code.py", line 4, in <module>
    ImportError: no module named 'simpleio'

Press any key to enter the REPL. Use CTRL-D to reload.
```

Tenemos un **ImportError**. Dice que no tenemos un módulo de nombre 'simpleio', o **no module named 'simpleio'**. ¡Este es el que acabamos de agregar en nuestro código!

Haga click en el enlace arriba para descargar el agrupado correcto. Extrae la carpeta **lib** del archivo descargado con el agrupado. Baja hasta ver el **simpleio.mpy**. ¡Este es el archivo de librería que andamos buscando! Sigue los pasos descritos arriba para cargar un archivo de librería de forma individual.

¡Ahora el LED comienza a parpadear de nuevo! Vamos a revisar la consola serial.



```
Press any key to enter the REPL. Use CTRL-D to reload.
soft reboot

Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.
code.py output:
```

¡Sin errores! Excelente. ¡Has resuelto correctamente un error de **ImportError** !

Si te topas con este error en el futuro, sigue los pasos descritos arriba y elige la librería que te hace falta.

Instalación de librerías en tarjetas que no son Express

Si tienes una Trinket M0 o una Gemma M0, vas a querer seguir los mismos pasos en el ejemplo anterior para instalar librerías mientras van haciendo falta. No siempre debes esperar a recibir el error `ImportError` dado que probablemente sabes que librería vas a necesitar ya que sabes que librería has agregado a tu código. Simplemente abre la carpeta **lib** que descargaste, escoge la librería que necesitas y arrástrala a la carpeta **lib** de tu unidad de disco **CIRCUITPY**.

Es posible que te quedes sin espacio en tu Trinket M0 o Gemma M0 incluso si vas copiando las librerías mientras las necesitas. Hay una serie de pasos que se pueden realizar para resolver este problema. Los vas a encontrar en la página de Depuración de la Guía de Aprendizaje para tu tarjeta.

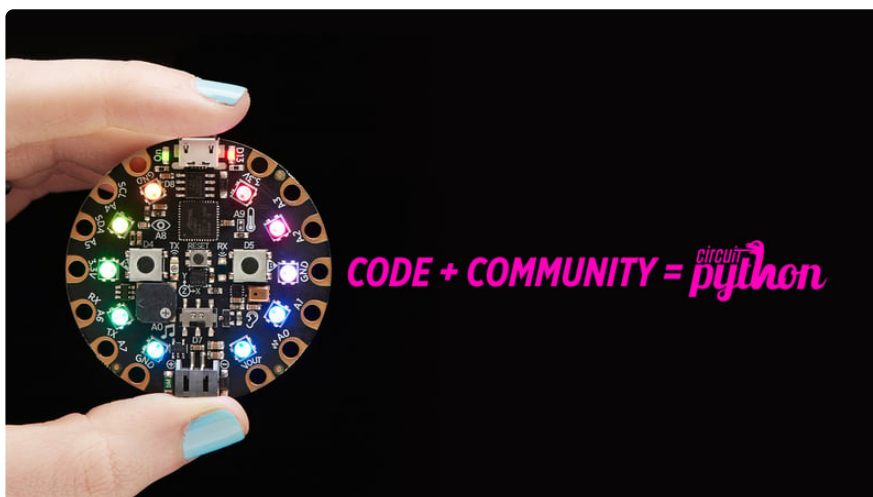
Actualizando Librerías o Ejemplos de CircuitPython

Las librerías y sus ejemplos se actualizan de forma constante, y es importante actualizar los archivos que tienes en tu unidad de disco **CIRCUITPY**.

Para actualizar las librerías de forma individual, sigue los mismos pasos descritos arriba. Cuando arrastras un archivo de librería a tu carpeta **lib**, te va a preguntar si la deseas reemplazar. Dile que sí. ¡Eso es todo!

Una nueva versión del agrupado de librerías es liberado cuando existe una actualización a una librería. Las actualizaciones incluyen cosas como arreglos a pulgas y nuevas características. Es importante revisar de vez en cuando para ver si las librerías que utilizas han sido actualizadas.

¡Bienvenido a la Comunidad!



¡Todos son bienvenidos! CircuitPython es Código Abierto. Esto significa que está disponible para todos para usar, editar, copiar y mejorar. Esto también significa que CircuitPython mejora porque tu eres una parte de él. No importa si es tu primer tarjeta microcontroladora o si eres un ingeniero informático, tu tienes algo importante que aportar a la Comunidad CircuitPython. ¡Vamos a subrayar algunos aspectos en los que te puedes involucrar!

The screenshot shows a Discord chat window for the #circuitpython channel. The chat history includes a message from Dan Halbert (posted at 2:35 PM) linking to a blog post about Adafruit Industries' new mu-editor-beta. The message includes a GitHub repository link and a photo of a Raspberry Pi board. Below the message, a user named cater responds with the path /home/halbert/.local/bin/mu. Another user, CGrover, thanks them. The chat interface includes a sidebar with channel names, a list of users, and a search bar at the top.

Existen muchos canales diferentes así que puedes escoger el que mejor se adapte a tus necesidades. Cada canal es mostrado en Discord como “#nombrecanal”. Existe el canal #projecthelp para asistencia con proyectos en progreso, o para buscar ayudas para el siguiente proyecto. Existe el canal #showandtell para mostrar tus últimas creaciones. ¡No tengas miedo de preguntar en ningún canal! Si estás incierto, el canal #general es un excelente lugar comenzar. Si hay otro canal que sea más apropiado para darte una mejor respuesta, alguien te va a guiar hacia él.

Page 51 of 97

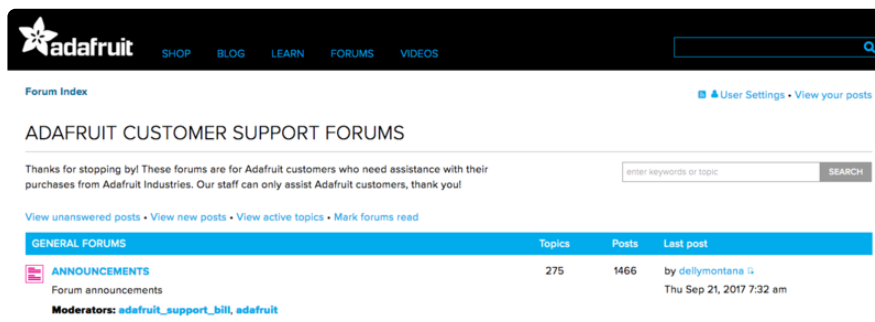
comentarios! Todos con cualquier nivel de experiencia son bienvenidos a participar en la conversación. ¡Nos encantaría escuchar lo que tienes que decir!

La mejor manera para contribuir con la comunidad es ayudar a otros en Discord. Ayudar a otros no siempre quiere decir responder sus preguntas. ¡Participa celebrando cuando algo sale bien! ¡Celebra tus errores! Algunas veces solo escuchar que alguien más ha tenido problemas en un área similar que por lo que uno está pasando puede ser suficiente para ayudarle a un maker a avanzar.

El Discord de Adafruit es un hackerspace abierto las 24x7, todos los días del año y adonde también puedes traer a tu nieta.

Visita <https://adafru.it/discord> () para crear una cuenta para Discord. ¡Estamos ansiosos por conocerte!

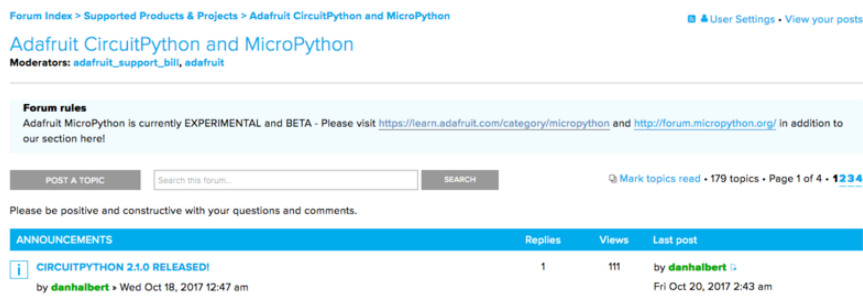
Foros de Adafruit



Los [Foros de Adafruit \(https://adafru.it/jlf\)](https://adafru.it/jlf) son el lugar perfecto para solicitar asistencia técnica. Adafruit tiene varias maravillosas personas a las que paga para dar asistencia técnica y responder cualquier pregunta que puedas tener. Ya sea que tu hardware te está dando problemas o que tu código parece no funcionar, los foros siempre están ahí para tus consultas. Usted necesita una cuenta de Adafruit para escribir a los foros. Usted puede utilizar la misma cuenta para realizar pedidos de Adafruit.

Mientras puede que en Discord encuentres una respuesta más rápida que en los foros, los foros son una fuente más confiable de información. Si quieres estar seguro de recibir una respuesta respaldada por Adafruit, los foros son el mejor lugar para ello.

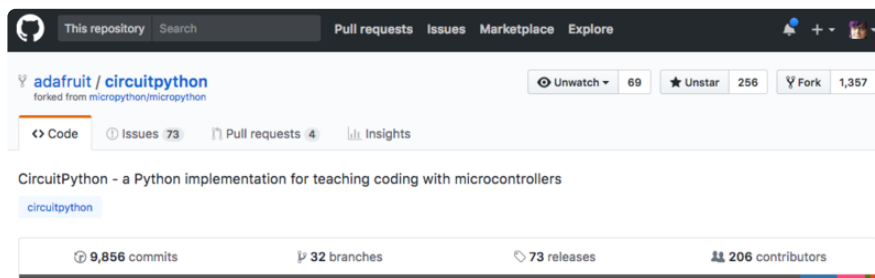
Las categorías de los foros cubren todo tipo de temas, incluyendo todo lo relacionado con Adafruit. La categoría de [Adafruit CircuitPython y MicroPython](#) () bajo "Supported Products & Projects" (o Productos y Proyectos con Asistencia Técnica) es el mejor lugar para enviar tus preguntas sobre CircuitPython.



Asegúrate de incluir todos los pasos que tomaste para llegar a donde te encuentras. Si involucra cableado, ¡envía una fotografía! Si tu código está teniendo problemas, ¡incluye tu código en la consulta! Estas son buenas prácticas para asegurarnos que vamos a tener suficiente información para ayudar con tu consulta.

Pueda que creas que apenas estás comenzando, pero definitivamente sabes algo que alguien más no. ¡Lo bueno de los foros es que tu también puedes ayudar a otros! Todos son bienvenidos is alentados a dar retroalimentación constructiva a las consultas enviadas. ¡Es una excelente forma para contribuir a la comunidad y compartir tus conocimientos!

Repositorios de Adafruit en GitHub



Ya sea que estés comenzando o que seas programador de toda la vida que te gustaría contribuir, hay formas para que todos sean parte de construir CircuitPython. GitHub es la mejor de las formas para contribuir a [CircuitPython \(\)](https://github.com/adafruit/circuitpython). Si necesitas una cuenta, visita [https://github.com/ \(\)](https://github.com/) y solicita la tuya.

Si eres nuevo hacia GitHub o programación en general, hay buenas oportunidades para ti. Visita [adafruit/circuitpython \(\)](https://github.com/adafruit/circuitpython) en GitHub, realiza un click en "[Issues \(\)](#)", y vas a encontrar una lista de problemas que han sido clasificados como "[good first issue \(\)](#)" (o buen primer problema). Estas son cosas que hemos identificado como algo en lo que cualquier con cualquier nivel de experiencia puede contribuir. Estos temas pueden incluir opciones como actualizar documentación, dar retroalimentación o arreglando pulgas sencillas.

❑	🔔	OneWire BusDevice driver	good first issue	2
		#338 opened 29 days ago by tannewt Long term		
❑	🔔	Feather M0 Adalogger does not have D8 or D7	good first issue	7
		#323 opened on Oct 13 by ladyada 3.0		
❑	🔔	Audit and fix native API for methods that accept and ignore extra args.	good first issue	
		#321 opened on Oct 12 by tannewt Long term		

¿Ya tienes experiencia y buscas un reto? Revisa el resto de problemas y vas a encontrar suficientes formas en las que puedes contribuir. Vas a encontrar de todo desde solicitudes para nuevas controladoras a actualización de módulos del núcleo. ¡Hay muchas oportunidades para todos los niveles!

Cuando se trabaja con CircuitPython, se pueden encontrar problemas. Si encuentras una pulga, ¡es excelente! ¡Nos encantan las pulgas! Enviar un reporte de problema detallado a GitHub es una manera invaluable para contribuir en la mejora de CircuitPython. Asegúrate de incluir los pasos relevantes para replicar el problema, así como cualquier otra información que te parezca relevante. Entre más detalle, ¡mejor!

Probar nuevo software es algo sencillo y de mucha ayuda. Simplemente cargue la nueva versión de CircuitPython o una librería en tu hardware para CircuitPython, y úsalo. Déjanos saber de cualquier problema que puedas encontrar, escribiendo un “issue” en GitHub. Probar software tanto en la versión estable como en las beta es una parte muy importante de contribuir a CircuitPython. ¡Nosotros no podemos encontrar todos los problemas! Necesitamos tu ayuda haciendo CircuitPython todavía mejor.

En GitHub, puedes enviar solicitud de características, dar retroalimentación, reportar problemas y mucho más. Si tienes consultas, ¡recuerda que Discord y los Foros están para ayudar!

ReadTheDocs

The screenshot shows the Adafruit CircuitPython documentation website. On the left is a sidebar with a search bar and a navigation menu. The main content area displays the 'audioio' module documentation, including a description, libraries, and a list of related modules like 'analogio' and 'audiobusio'.

[ReadTheDocs \(https://adafru.it/Beg\)](https://adafru.it/Beg) es un excelente recurso para una mirada más profunda a CircuitPython. Aquí es donde vas a encontrar cosas como documentación del API y detalles sobre los módulos del núcleo. También hay Guías de Diseño que incluyen guías sobre como contribuir a CircuitPython.

RTD nos da un acceso a bajo nivel de CircuitPython. Hay detalles sobre cada uno de los [módulos núcleo \(\)](#). Cada módulo lista las librerías disponibles. Cada página de módulo de librería lista los parámetros disponibles y la explicación de cada uno. En muchos casos, vas a encontrar ejemplos rápidos de código que te van a ayudar a entender como funcionan los módulos y sus parámetros, sin embargo no va a tener explicaciones detalladas como las Guías de Aprendizaje. Si quieres ayuda aprendiendo que sucede detrás del telón en el código de CircuitPython que escribes, ¡ReadTheDocs está para ayudar!

Here is blinky:

```
import digitalio
from board import *
import time

led = digitalio.DigitalInOut(D13)
led.direction = digitalio.Direction.OUTPUT
while True:
    led.value = True
    time.sleep(0.1)
    led.value = False
    time.sleep(0.1)
```

Consola Serial Avanzada en Windows Controlador para Windows 7

Si estás utilizando Windows 7, usa el enlace abajo para descargar el paquete con el controlador. No vas a tener que instalar controladores ni para Mac, Linux o Windows 10.

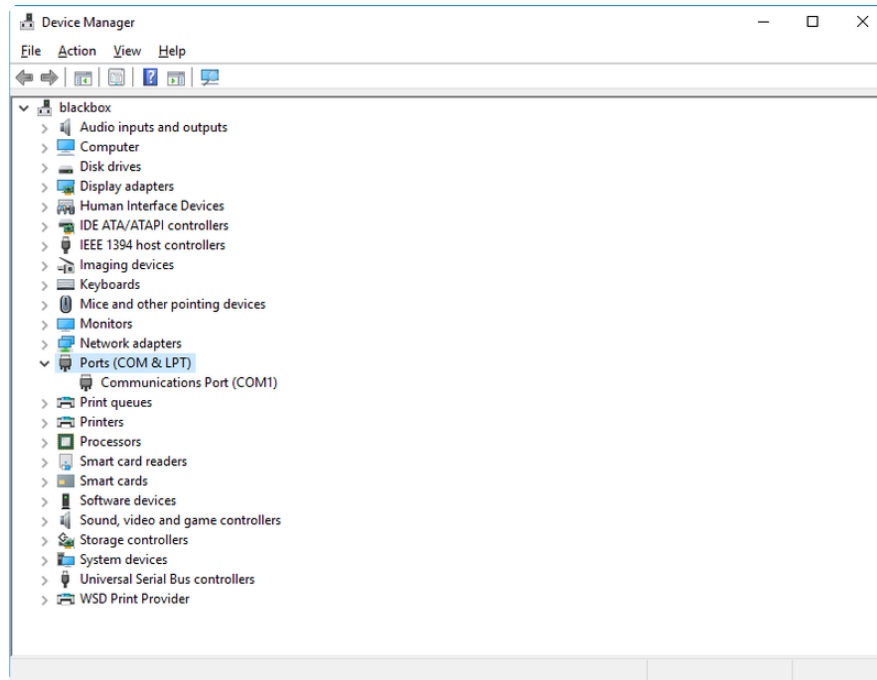
**Descarga Controladores para
Windows 7**

<https://adafru.it/AB0>

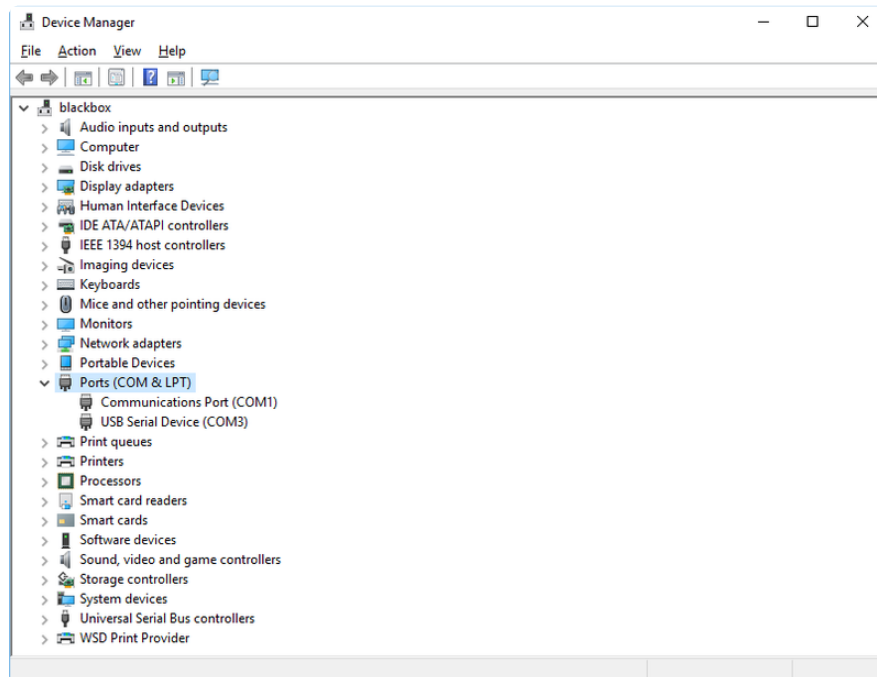
¿Que es el COM?

Primero, quieres saber cual de los puertos seriales es el que usa tu tarjeta. Cuando conectas tu tarjeta por USB a tu computadora, se conecta a un puerto serial. Este puerto es un acceso por el cual tu tarjeta se comunica con tu computadora utilizando USB.

Nosotros vamos a utilizar el Administrador de Dispositivos de Windows para determinar cual puerto de la tarjeta estás utilizando. La forma más sencilla para determinar cual puerto está utilizando la tarjeta es primero revisar la lista de puertos **sin** haber conectado la tarjeta. Abre el Administrador de Dispositivos. De click a Puertos (COM y LPT). Deberías encontrar algo en esta lista con un nombre como COM# donde # es un número.



Ahora conecta tu tarjeta. La lista del Administrador de Dispositivos se va a refrescar y va a aparecer un puerto nuevo bajo Puertos (COM & LPT). Vas a encontrar un nuevo (COM#) en esta lista.



Algunas veces el puerto se va a referir al nombre de la tarjeta. Otras veces puede que se llame algo como USB Serial Device, como se observa en la imagen arriba. De cualquier forma, aparece un nuevo (COM#) seguido por el nombre. Este es el puerto de la tarjeta que estás utilizando.

Instalando Putty

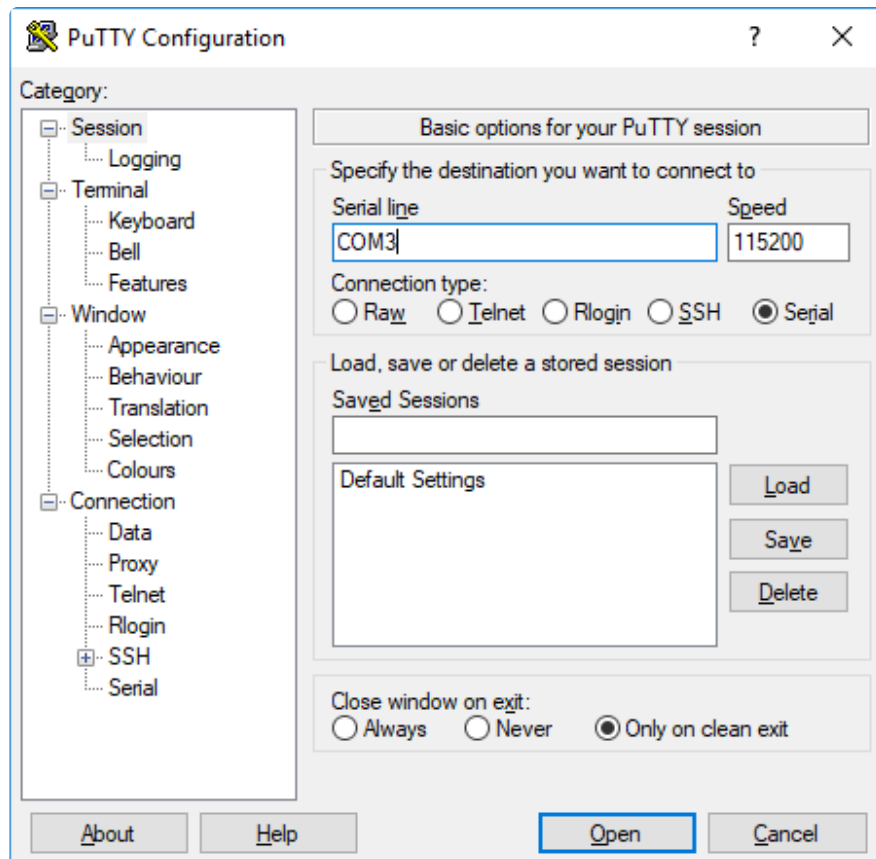
Si estás utilizando Windows, vas a tener que descargar una aplicación terminal. Vamos a utilizar PuTTY.

Lo primero que debes hacer es descargar la [última versión de PuTTY \(\)](#). Vas a querer descargar el archivo instalador para Windows. Lo más probable es que vas a necesitar la versión de 64-bits. Descargar el archivo e instalar la aplicación en tu computadora. Si tienes problemas, puedes tratar descargando más bien la versión de 32-bits. Sin embargo, la versión de 64-bits va a funcionar en la mayoría de computadoras personales.

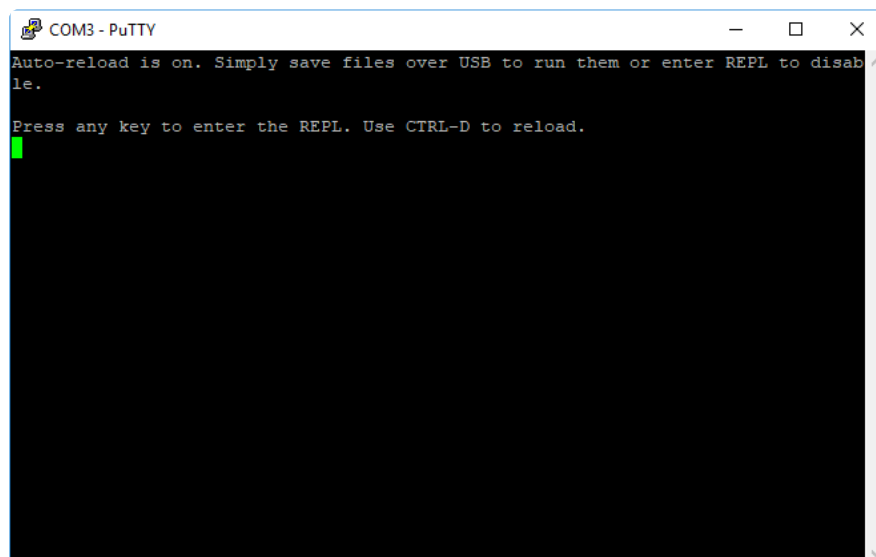
Ahora va a querer abrir PuTTY.

- Bajo **Connection type**: busca el botón a la par de **Serial**.
- En la caja bajo **Serial line**, digita el puerto serial que encuentre que utiliza tu tarjeta.
- En la caja bajo **Speed**, digita 115200. Esto se llama la tasa de baudios, que es la velocidad en bits por segundo que los datos se transfieren por la conexión serial. Para tarjetas con USB integrado no importa mucho, pero para tarjetas como las ESP8266 y otras con un chip adicional, la velocidad requerida por la placa es 115200 bits por segundo. ¡Así que por qué no usar 115200!

Si deseas guardar estos parámetros, utiliza las opciones bajo **Load, save or delete a stored session**. Digita el nombre de la sesión en la caja bajo **Saved Sessions**, y dá click en el botón **Save** a la derecha.



Una vez que los parámetros se han digitado, estás listo para conectarte a la consola serial. Realice un click a “Open” en la parte de abajo de la ventana. Una nueva ventana se abrirá.



Si no hay código en ejecución, la ventana va a verse en blanco o se va a ver como la ventana arriba. Ahora estás listo para ver los resultados de tu código.

¡Buen trabajo! ¡Te has conectado a la consola serial!

Consola Serial Avanzada en Mac y Linux

Conectándose a la consola serial de Mac y Linux es esencialmente el mismo proceso. Ninguno de los dos sistemas operativos necesita que se instalen controladores. En MacOSX, la aplicación **Terminal** viene instalada. En Linux hay una variedad como `gnome-terminal` (llamada Terminal) o Konsole bajo KDE.

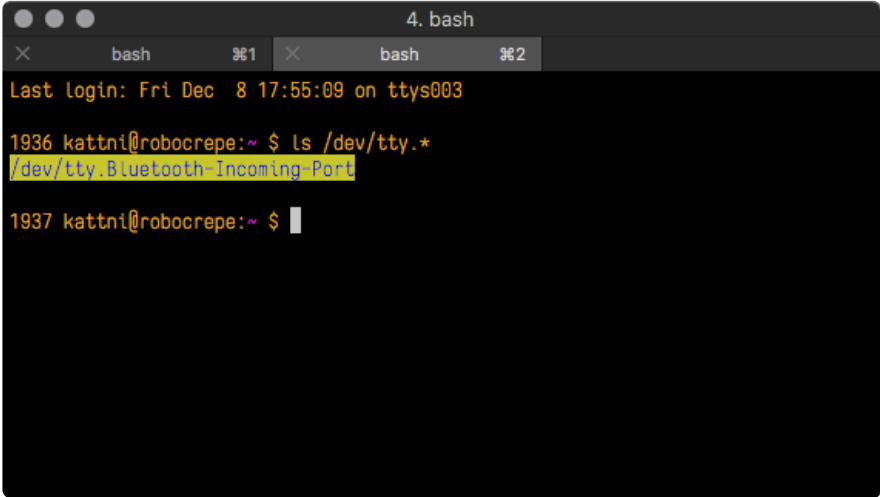
¿Cual es el puerto?

Primero vas a querer averiguar cual es el puerto serial que utiliza tu tarjeta. Cuando conectas tu tarjeta por USB a tu computadora, se conecta a un puerto serial. El puerto es un acceso por el cual tu tarjeta se comunica con tu computadora utilizando USB.

Vamos a utilizar Terminal para determinar cual puerto está usando la tarjeta. La forma más sencilla para determinar cual puerto está utilizando la tarjeta es primero revisar **sin** haber conectado la tarjeta. En Mac, abre Terminal, y escribe lo siguiente:

```
ls /dev/tty.*
```

Cada conexión serial aparece en la carpeta `/dev/`. Tiene un nombre que comienza con `tty`. El comando `ls` muestra una lista del contenido de la carpeta. Puede utilizar `*` como un metacaracter, para buscar archivos que comienzan su nombre con las mismas letras, pero terminan en algo diferente. En este caso, te pedimos que veas todo lo listado en `/dev/` que comience con `tty`. y termine en cualquier cosa. Esto nos va a mostrar las conexiones seriales presentes.

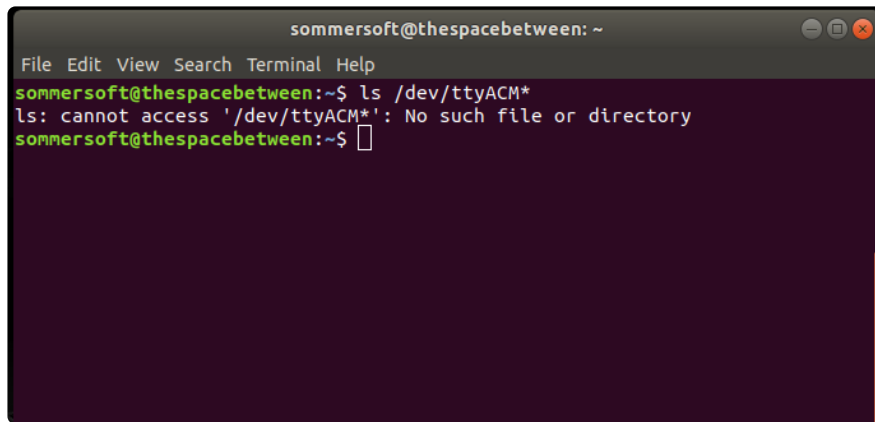


```
4. bash
bash %1 bash %2
Last login: Fri Dec 8 17:55:09 on ttys003
1936 kattni@robocrepe:~ $ ls /dev/tty.*
/dev/tty.Bluetooth-Incoming-Port
1937 kattni@robocrepe:~ $
```

Para Linux, el procedimiento es el mismo, sin embargo, el nombre es ligeramente diferente. Si está utilizando Linux, vas a escribir:

```
ls /dev/ttyACM*
```

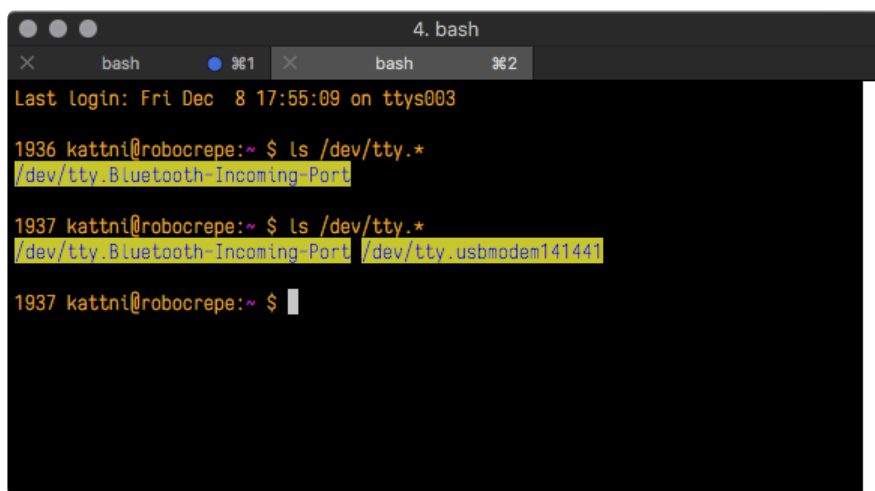
El concepto es el mismo con Linux. Te pedimos que veas lo listado de la carpeta `/dev/`, comenzando con `ttyACM` y terminando en cualquier cosa. Esto va a mostrar las conexiones seriales presentes. En el ejemplo anterior el error es indicando que no hay conexiones seriales presentes que comiencen con `ttyACM`.

A terminal window titled 'sommersoft@thespacebetween: ~' with a menu bar (File, Edit, View, Search, Terminal, Help). The command 'ls /dev/ttyACM*' is entered, and the output is 'ls: cannot access '/dev/ttyACM*': No such file or directory'. The prompt is 'sommersoft@thespacebetween:~\$'.

Ahora conecta tu tarjeta. Utilizando Mac, escribe:

```
ls /dev/tty.*
```

Esto va a mostrar las conexiones seriales presentes, en las cuales se incluye la de tu tarjeta.

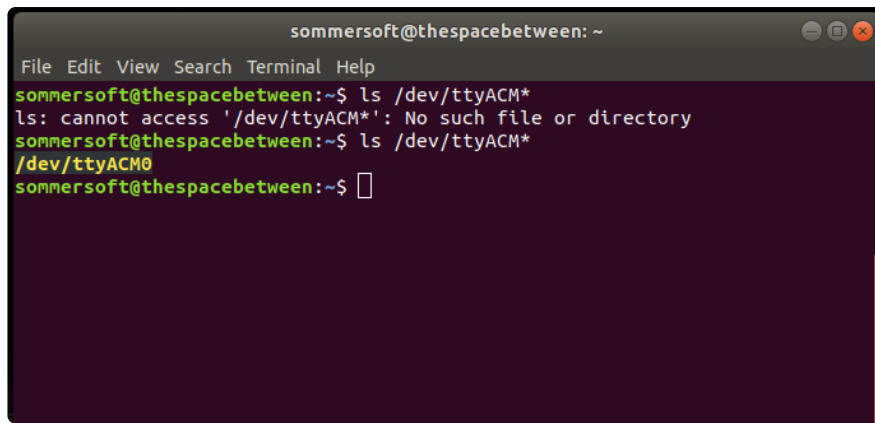
A terminal window titled '4. bash' with two tabs labeled 'bash' and 'bash'. The first tab shows 'Last login: Fri Dec 8 17:55:09 on ttys003'. The second tab shows the command 'ls /dev/tty.*' being executed twice. The first execution shows '/dev/tty.Bluetooth-Incoming-Port'. The second execution shows '/dev/tty.Bluetooth-Incoming-Port' and '/dev/tty.usbmodem141441'. The prompt is 'kattni@robocrepe:~\$'.

Utilizando Mac, un nuevo puerto aparece con el nombre `/dev/tty.usbmodem141441`. La parte `tty.usbmodem141441` es en este ejemplo el nombre de la tarjeta que estás usando. La tuya va a tener un nombre similar.

Utilizando Linux, escribe:

```
ls /dev/ttyACM*
```

Esto va a mostrar las conexiones seriales presentes, en las cuales se incluye la de tu tarjeta.



```
sommersoft@thespacebetween: ~  
File Edit View Search Terminal Help  
sommersoft@thespacebetween:~$ ls /dev/ttyACM*  
ls: cannot access '/dev/ttyACM*': No such file or directory  
sommersoft@thespacebetween:~$ ls /dev/ttyACM*  
/dev/ttyACM0  
sommersoft@thespacebetween:~$
```

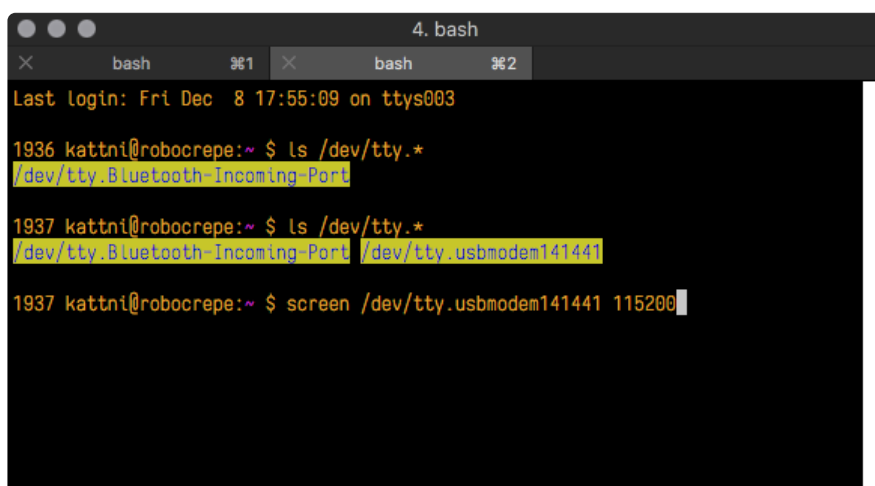
Utilizando Linux, un nuevo puerto aparece con el nombre `/dev/ttyACM0`. La parte `ttyACM0` en este ejemplo es el nombre de la tarjeta que estás utilizando. La tuya va a tener un nombre similar.

Conectándose con screen

Ahora que conoces el nombre de la tarjeta que estás utilizando, estás listo para conectarte a la consola serial. Vamos a utilizar un comando llamado `screen`. El comando `screen` se incluye con MacOS. Los usuarios de Linux es posible que lo vayan a tener que instalar utilizando su administrador de paquetes. Para conectarse a la consola serial, utilice Terminal. Escriba el siguiente comando, reemplazando `nombre_tarjeta` por el nombre que encontraste que usa tu tarjeta:

```
screen /dev/tty.board_name 115200
```

La primera porción de este comando establece que vas a utilizar el comando `screen`. La segunda parte le dice a `screen` el nombre de la tarjeta que estás tratando de usar. La tercera parte le indica a `screen` la tasa de baudios a utilizar para la conexión serial. La tasa de baudios es la velocidad en bits por segundo en que se transfieren datos por la conexión serial. En este caso la velocidad requerida por la tarjeta es 115200 bits por segundo.



```
4. bash  
Last login: Fri Dec 8 17:55:09 on ttys003  
1936 kattrni@robocrepe:~$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port  
1937 kattrni@robocrepe:~$ ls /dev/tty.*  
/dev/tty.Bluetooth-Incoming-Port /dev/tty.usbmodem141441  
1937 kattrni@robocrepe:~$ screen /dev/tty.usbmodem141441 115200
```

```
sommersoft@thespacebetween: ~  
File Edit View Search Terminal Help  
sommersoft@thespacebetween:~$ ls /dev/ttyACM*  
ls: cannot access '/dev/ttyACM*': No such file or directory  
sommersoft@thespacebetween:~$ ls /dev/ttyACM*  
/dev/ttyACM0  
sommersoft@thespacebetween:~$ screen /dev/ttyACM0 115200
```

Digita enter para ejecutar el comando. Se abre en la misma ventana. Si ningún código está corriendo, la ventana se va a ver en blanco. De otra forma, vas a ver la salida de tu código.

¡Buen trabajo! ¡Te has conectado a la consola serial!

Permisos en Linux

Si tratas de conectarte con `screen` y no funciona, es posible que tengas un problema con los permisos. Linux le da seguimiento a los usuarios y grupos y lo que pueden hacer y no hacer, como acceso al hardware asociado con la consola serial para correr `screen`. Así que si ves algo como esto:

```
ackbar@desk: ~  
ackbar@desk:~$ screen /dev/ttyACM0  
[screen is terminating]  
ackbar@desk:~$
```

es posible que necesites darte acceso. Existen básicamente dos formas para lograrlo. La primera es ejecutar `screen` con el comando de `sudo`, el cual temporalmente entrega privilegios elevados.

```
ackbar@desk: ~  
ackbar@desk:~$ screen /dev/ttyACM0  
[screen is terminating]  
ackbar@desk:~$ sudo screen /dev/ttyACM0  
[sudo] password for ackbar:
```

Una vez que escribas tu contraseña, ya debería funcionar:

```
ackbar@desk: ~  
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.  
  
Press any key to enter the REPL. Use CTRL-D to reload.  
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit Trinket M0 with samd21e18  
>>>
```

La segunda forma es agregándote al grupo asociado con el hardware. Para averiguar cual grupo es el correcto, utiliza el comando `ls -l` como se muestra abajo. El nombre del grupo está marcado en un círculo rojo.

Ahora utiliza el comando `adduser` para agregar a tu usuario al grupo. Necesitas permisos elevados para realizar esto, por lo cual necesitas utilizar `sudo`. En el ejemplo abajo, el grupo es `adm` y el usuario `ackbar`.

```
ackbar@desk: ~  
ackbar@desk:~$ ls -l /dev/ttyACM0  
crw-rw---- 1 root adm 166, 0 Dec 21 08:29 /dev/ttyACM0  
ackbar@desk:~$ sudo adduser ackbar adm  
Adding user `ackbar' to group `adm' ...  
Adding user ackbar to group adm  
Done.  
ackbar@desk:~$
```

Luego de que te agregues al grupo, necesitas deslogarte y logearte de nuevo, o en algunos casos, reiniciar tu computadora. Luego de que te logees de nuevo, verifica que te has agregado utilizando el comando `groups`. Si no estás en el grupo, reinicia y revisa de nuevo.

```
ackbar@desk: ~  
ackbar@desk:~$ groups  
ackbar adm sudo  
ackbar@desk:~$
```

Ahora debes poderte conectar con `screen` sin necesidad de `sudo`.

```
ackbar@desk: ~  
ackbar@desk:~$ groups  
ackbar adm sudo  
ackbar@desk:~$ screen /dev/ttyACM0 115200
```

Y ya estás adentro:

```
ackbar@desk: ~  
Auto-reload is on. Simply save files over USB to run them or enter REPL to disable.  
  
Press any key to enter the REPL. Use CTRL-D to reload.  
Adafruit CircuitPython 2.1.0 on 2017-10-17; Adafruit Trinket M0 with samd21e18  
>>> |
```

Los ejemplos de arriba utilizan `screen` , pero si lo prefieres puede utilizar otros programas como `putty` o `picocom` .

PyCharm y CircuitPython

ESTE PROCESO NO FUNCIONA. Este plugin nunca fue diseñado para trabajar con CircuitPython. El proceso descrito es una forma de lograrla con funcionalidad limitada. SI DESEA UTILIZAR REPL CON PYCHARM, UTILICE LA TERMINAL INTEGRADA Y `screen`.

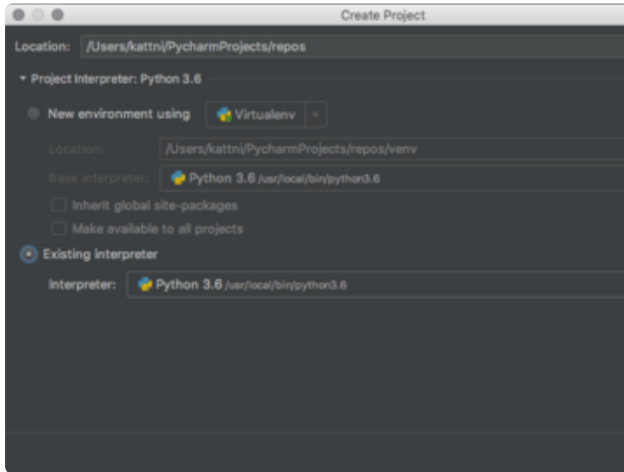
[PyCharm](https://adafru.it/xNC) (<https://adafru.it/xNC>) es un editor con todas las características incluyendo cosas super útiles como completado de código y señalado de errores. Está disponible gratis en su versión comunitaria.



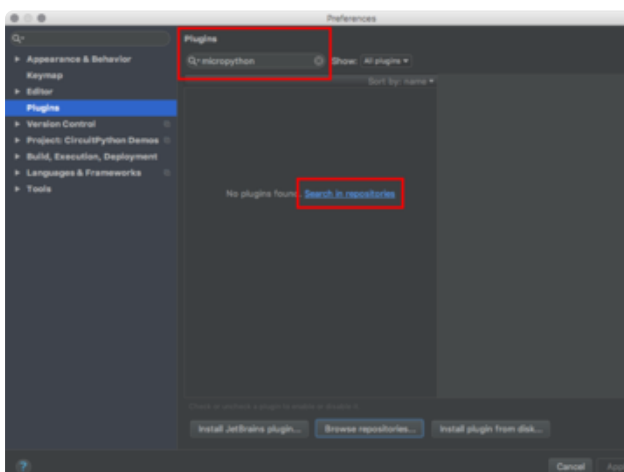
Recientemente, agregaron un plugin para [MicroPython](#) () el cual habilita el uso del REPL desde el editor.

CircuitPython no está bajo mantenimiento oficial, sin embargo ¡tenemos algunos pasos para lograr que funcione!

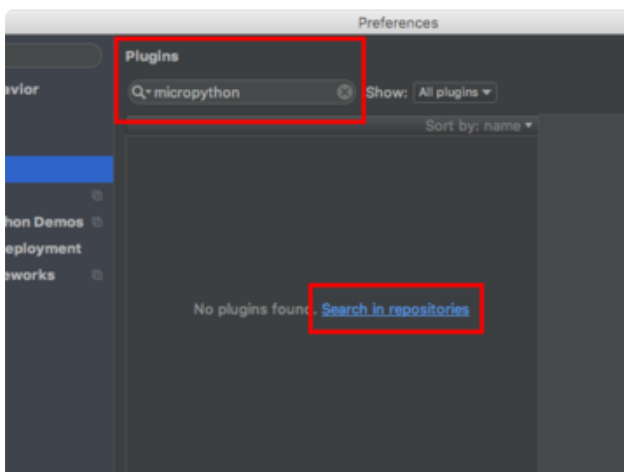
[Descargue](#) () e instale PyCharm en su computadora. Ahora, ¡conecte la tarjeta y siga los pasos descritos arriba!



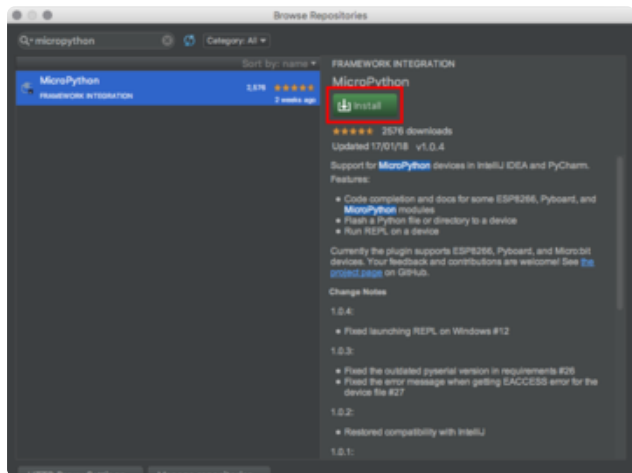
Cree un nuevo proyecto o abra un proyecto existente.



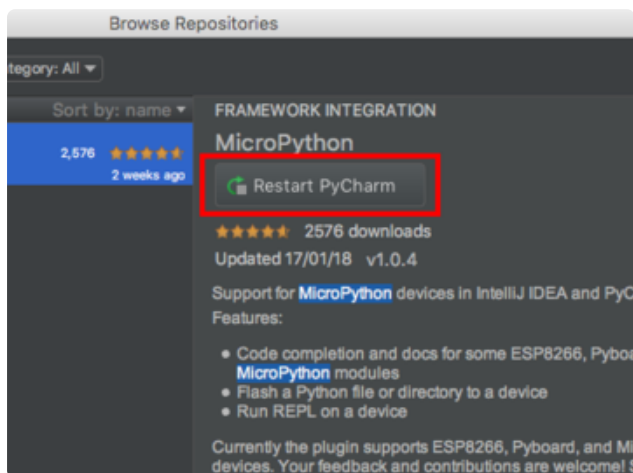
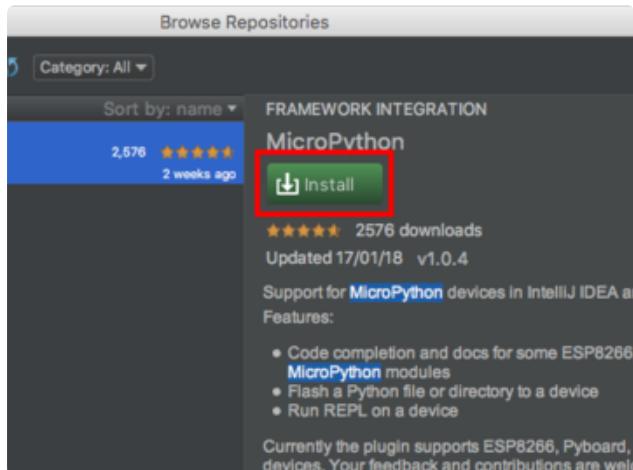
Abra PyCharm y luego Preferences/ Settings. Realice un click en **Plugins** y busque “micropython”. Realice un click en **Search in repositories**



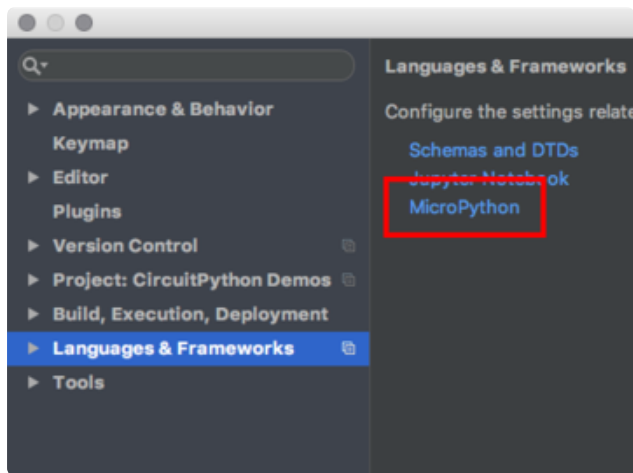
ESTE PROCESO NO FUNCIONA. Este plugin nunca fue diseñado para trabajar con CircuitPython. El proceso descrito es una forma de lograrla con funcionalidad limitada. SI DESEA UTILIZAR REPL CON PYCHARM, UTILICE LA TERMINAL INTEGRADA Y screen.



Realice un click en **Install**.

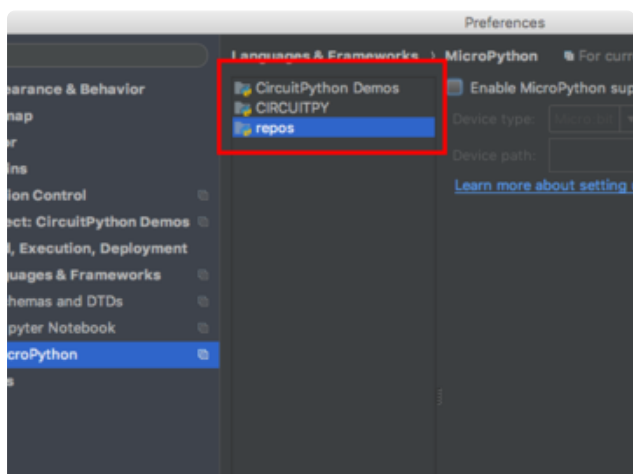


Una vez que el proceso completa, de click en **Restart PyCharm**.

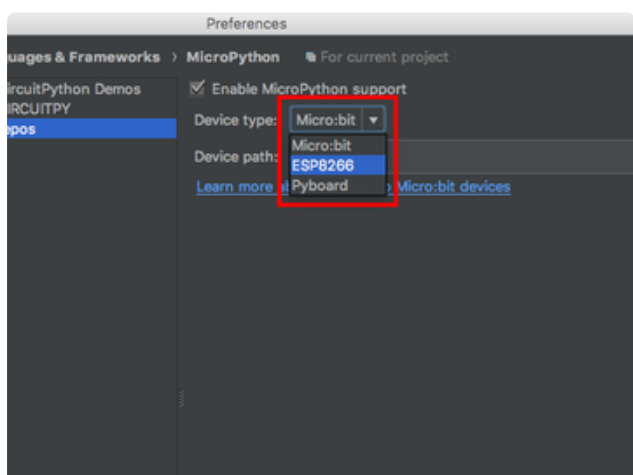


Una vez reiniciado, abra Preferences/ Settings. Dé click en **Languages & Frameworks** y seleccione **MicroPython**.

ESTE PROCESO NO FUNCIONA. Este plugin nunca fue diseñado para trabajar con CircuitPython. El proceso descrito es una forma de lograrla con funcionalidad limitada. SI DESEA UTILIZAR REPL CON PYCHARM, UTILICE LA TERMINAL INTEGRADA Y screen.

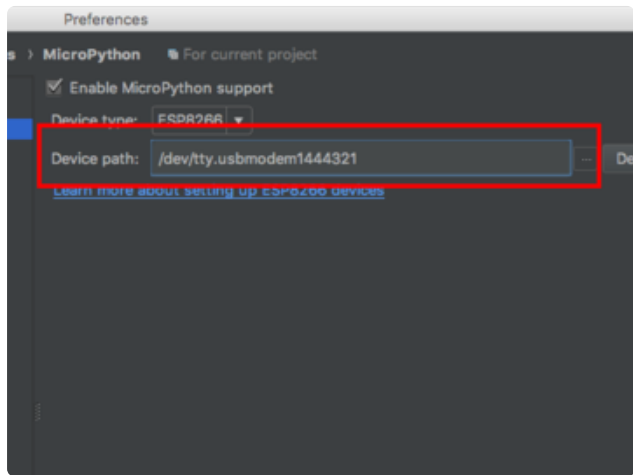


Escoja su carpeta de proyecto de la lista.



Escoja **ESP8266** de la lista de Device Names (o Nombres de Dispositivos).

Por ahora, vas a utilizar esta opción sin importar la tarjeta que utilices.

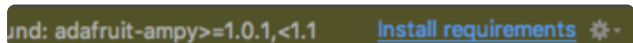


Vas a necesitar agregar manualmente la ruta del dispositivo o **Device Path**. Esta es la ruta hacia tu conexión serial. Una vez escrita, realice un click en **Ok**.

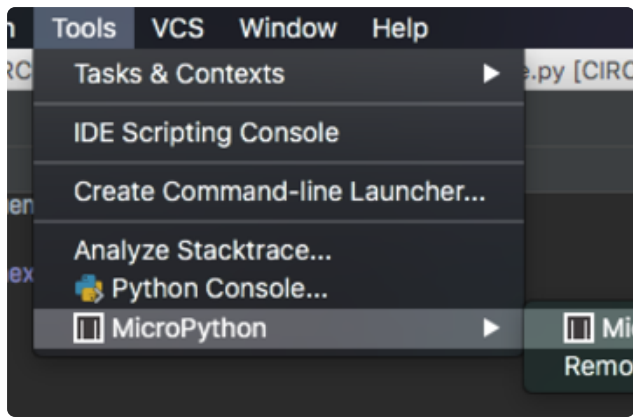
Si necesitas ayuda encontrando la consola serial de tu dispositivo visita la sección [Advanced Serial Console on Windows \(\)](#) y [Advanced Serial Console on Mac and Linux \(\)](#).

ESTE PROCESO NO FUNCIONA. Este plugin nunca fue diseñado para trabajar con CircuitPython. El proceso descrito es una forma de logarla con funcionalidad limitada. SI DESEA UTILIZAR REPL CON PYCHARM, UTILICE LA TERMINAL INTEGRADA Y screen.

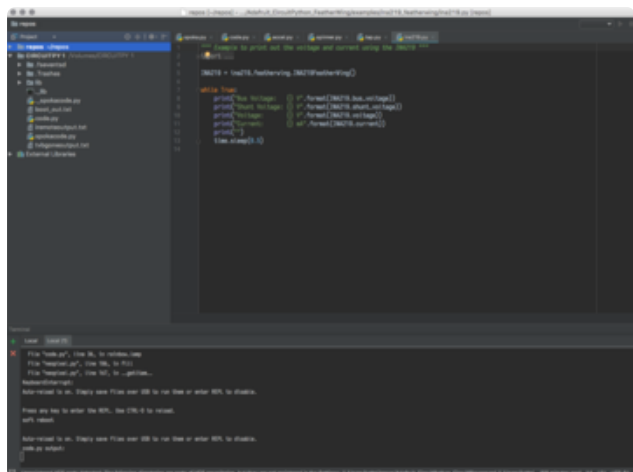
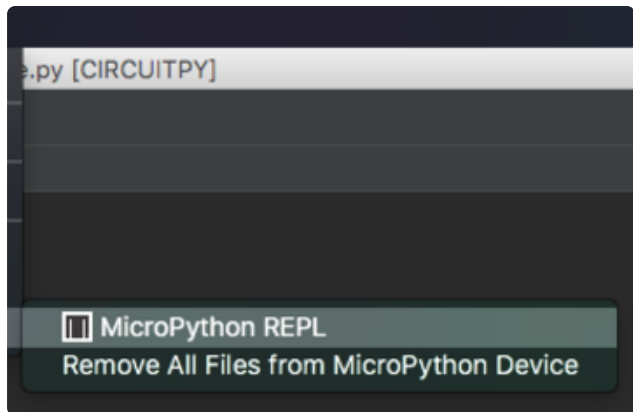
Ahora, abra cualquier archivo de Python de la carpeta de proyecto que has seleccionado.



Puede que aparezca un mensaje diciendo "Packages required for ESP8266 support not found", seguidos de una lista de paquetes. Haga click en **Install Requirements** en este mensaje para instalar los paquetes requeridos.



Ahora le puedes dar click al menú de **Tools** (o Herramientas) y vas a poder encontrar un menú **MicroPython**. Señala esto, y vas a encontrar el **REPL** para **MicroPython** disponible.



El REPL se abrirá en la parte inferior de la ventana de PyCharm. ¡Ya puedes comenzar a programar!

Si deseas utilizar el REPL con una tarjeta diferente, vas a tener que ir a modificar la ruta del dispositivo a la ruta apropiada para la tarjeta que deseas utilizar.

Para cambiar la ruta de dispositivo, siga los pasos descritos arriba, comenzando por agregar manualmente la ruta de dispositivo.

El REPL de PyCharm no va a trabajar con múltiples tarjetas al mismo tiempo. Seguir los pasos descritos arriba para múltiples proyectos no resulta para abrir múltiples conexiones a REPL de forma simultánea.

Las funcionalidades de "Remove All Files from MicroPython Device" y "Flash Project" no funcionan, pero podrían corromper tu tarjeta. No trates de utilizar estas funcionalidades. Recuerda, CircuitPython no está bajo mantenimiento oficial, y estos son solo algunos pasos para lograr que funcione.

ESTE PROCESO NO FUNCIONA. Este plugin nunca fue diseñado para trabajar con CircuitPython. El proceso descrito es una forma de lograrla con funcionalidad limitada. SI DESEA UTILIZAR REPL CON PYCHARM, UTILICE LA TERMINAL INTEGRADA Y screen.

CircuitPython para la ESP8266

Nosotros ya no estamos dando mantenimiento a CircuitPython en ESP8266. Esta página es solamente por razones históricas. No hay garantía que las instrucciones continúen funcionando.

¿Porqué no damos mantenimiento a las ESP8266?

Correr CircuitPython en las ESP8266 no ha sido una buena experiencia para los usuarios. Es difícil enviar archivos al dispositivo porque no tiene USB nativo, y rápidamente te puedes quedar sin memoria RAM (queda menos de lo que crees una vez que activas los componentes para redes). Hemos decidido utilizar las ESP solo como un co-procesador. Específicamente la ESP32 porque tiene muy buenas capacidades de TLS/SSL las cuales son esenciales para las interacciones más básicas.

Si desea utilizar ESP8266, favor mantenga la versión 3.x teniendo en mente que no le damos mantenimiento. ¡También puede usar MicroPython para ESP8266 a la cual le dan mantenimiento!

[Si deseas agregar capacidades de WiFi, revisa nuestra guía sobre usar las ESP32/ESP8266 como un co-procesador \(\)](#).

Sobre ESP8266 para CircuitPython (3.x)

Tenemos dos sub-versiones de CircuitPython, la primaria es para tarjetas basadas en ATSAMD21/51 que tienen conectividad nativa por USB. USB nativo significa que la

tarjeta aparece como una unidad de disco llamada **CIRCUITPY** donde puedes almacenar tus archivos.

También hay CircuitPython para tarjetas como las ESP8266 y nRF52832, estos son muy buenos chips con WiFi y Bluetooth integrados, respectivamente. **¡Pero no tienen soporte nativo para USB!** Esto significa que no hay forma que el chip aparezca como una unidad de disco. Usted todavía las puede usar con CircuitPython pero es mucho más difícil, así que no las recomendamos para principiantes. Esto es lo que debes saber si utilizas chips no-nativos a CircuitPython:

- ¡Solo tienes una conexión al REPL! No hay capacidades de teclado y mouse HID u otras interfaces USB.
- No hay unidades de disco para arrastrar archivos, y los archivos deben ser manipulados con una herramienta como **ampy** la cual “escribe” tu archivo usando el REPL.
- Cargar CircuitPython requiere herramientas de línea de comandos

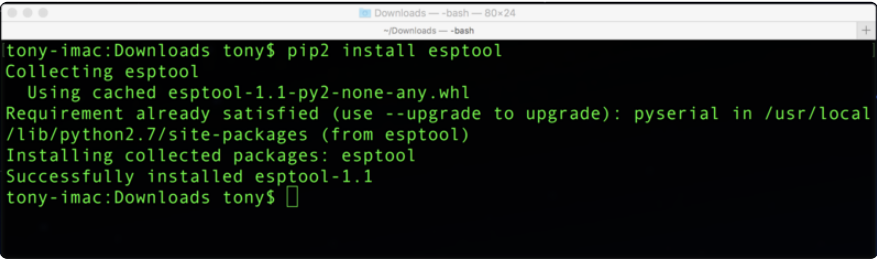
Instalando CircuitPython en una ESP8266

Para utilizar CircuitPython con una ESP8266 primero necesitas cargarlo con el último firmware.

Descargue esptool

Primero instale la herramienta **esptool.py** que permite subirle firmware a una ESP8266. La forma más sencilla para instalar esta herramienta es con el administrador de paquetes para Python llamado pip . Si aún no lo tiene, debe [instalar Python 2.7 \(\)](#) (asegúrese de marcar la caja para poner Python en tu sistema cuando instalas Windows) y luego ejecute el siguiente comando en una terminal: **pip install esptool**

Tome nota que en Mac OSX y Linux puede que necesite ejecutar el comando en conjunto con sudo, de esta manera: **sudo pip install esptool**



```
tony-imac:Downloads tony$ pip2 install esptool
Collecting esptool
  Using cached esptool-1.1-py2-none-any.whl
Requirement already satisfied (use --upgrade to upgrade): pyserial in /usr/local/lib/python2.7/site-packages (from esptool)
Installing collected packages: esptool
Successfully installed esptool-1.1
tony-imac:Downloads tony$
```

Si recibes un error de que `esptool.py` solo trabaja con Python 2.x, trata de ejecutarlo de nuevo pero con `pip2` en lugar de `pip` (lo más probable es que tu sistema esté usando Python 3 y el comando `pip` se confunde con cual usar).

Descargue la última versión del firmware de CircuitPython

Ahora, descargue el archivo de la última versión del firmware de CircuitPython para ESP8266:

Descargue el último firmware para
la ESP8266 Huzzah (v3.1.2)

<https://adafru.it/F9z>

Alistando la ESP8266 para booteo

Ahora necesitas poner a la ESP8266 en modo de subida de firmware. En cada ESP8266 es un poco diferente:

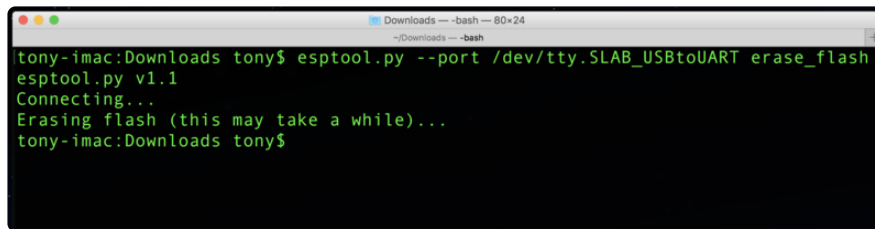
- **Para un módulo ESP8266 puro**, necesitas cablear botones hacia tierra para los pines de GPIO0 y RESET. Presionas el botón de GPIO0 (o conectas la línea a tierra) y mientras lo mantienes presionado, presiona y suelta el botón RESET (o conectas y desconectas la línea a tierra), y luego sueltas GPIO0.
- **Para las [HUZZAH ESP8266 breakout](http://adafru.it/2471)** (<http://adafru.it/2471>) tiene botones de GPIO0 y RESET integrados en la tarjeta. Presiona GPIO0, luego presiona y suelta RESET (manteniendo presionado GPIO0) y luego suelta GPIO0.
- **Para la [ESP8266 Huzzah en formato Feather](#)** () no necesitas hacer nada especial para activar el modo de subida de firmware. Esta tarjeta está construida para detectar cuando el puerto serial está abierto para subida y automáticamente configura el módulo de la ESP8266 para recibir el firmware. ¡Asegúrate de [instalar el controlador para los SiLabs CP210x](#) () en Windows y en Mac OSX para permitir que se vea el puerto de la tarjeta! En Windows quieres el controlador normal VCP, y no el controlador con enumeración serial ('with Serial Enumeration').

Borrar la ESP8266

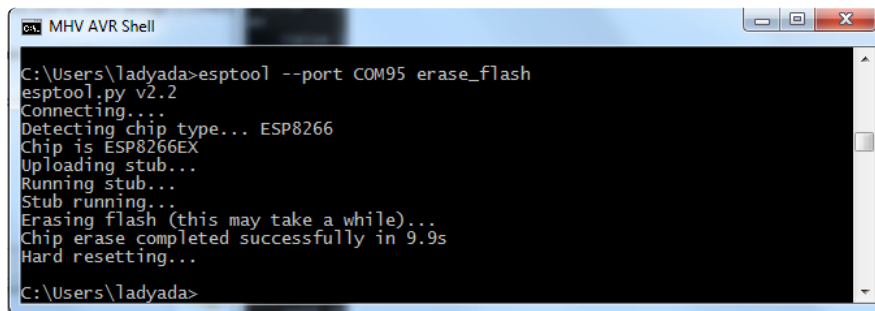
Es recomendable eliminar la memoria de flash completa de la tarjeta ESP8266 antes de subirle firmware. Ejecute el siguiente comando en una terminal para ejecutar este borrado:

```
esptool.py --port NOMBREPUERTO_ESP8266 erase_flash
```


Donde **NOMBREPUERTO_ESP8266** es la ruta o el nombre del puerto serial que está conectado a la ESP8266. El nombre exacto varía dependiendo del tipo de convertidor serial así que vas a querer revisar la lista de puertos seriales con y sin la tarjeta conectada para encontrar su nombre.



```
tony-imac:Downloads tony$ esptool.py --port /dev/tty.SLAB_USBtoUART erase_flash
esptool.py v1.1
Connecting...
Erasing flash (this may take a while)...
tony-imac:Downloads tony$
```



```
C:\Users\ladyada>esptool --port COM95 erase_flash
esptool.py v2.2
Connecting...
Detecting chip type... ESP8266
Chip is ESP8266EX
Uploading stub...
Running stub...
Stub running...
Erasing flash (this may take a while)...
Chip erase completed successfully in 9.9s
Hard resetting...
C:\Users\ladyada>
```

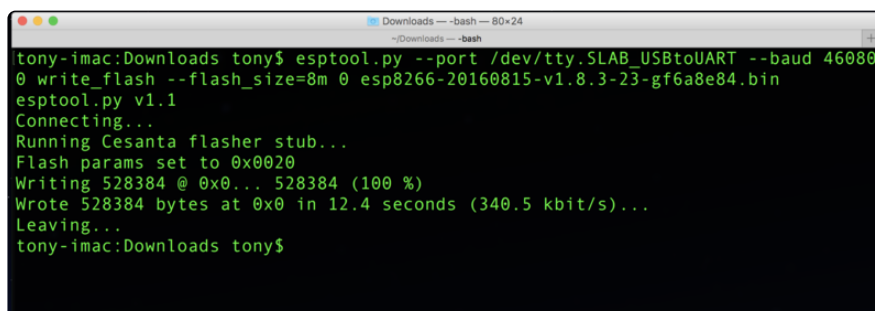
Programar la ESP8266

Ahora, activa de nuevo el modo de subida de firmware en la ESP8266 y ejecuta el siguiente comando para cargarle el archivo de firmware:

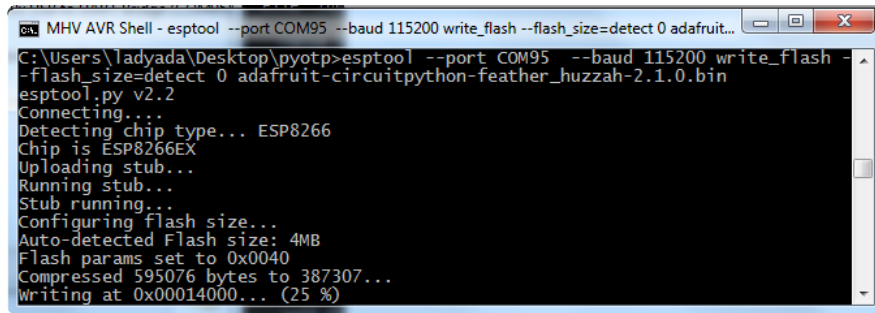
```
esptool.py --port NOMBREPUERTO_ESP8266 --baud 115200 write_flash --flash_size=detect 0 firmware.bin
```

Again set **ESP8266_PORTNAME** to the path or name of the serial port that is connected to the ESP8266. In addition set **firmware.bin** to the name or path to the firmware file you would like to load.

De nuevo, **NOMBREPUERTO_ESP8266** es la ruta o nombre del puerto serial que está conectado a la ESP8266. Así mismo, defina el nombre o ruta del archivo de firmware que desea cargar en lugar de **firmware.bin**.



```
tony-imac:Downloads tony$ esptool.py --port /dev/tty.SLAB_USBtoUART --baud 460800 write_flash --flash_size=8m 0 esp8266-20160815-v1.8.3-23-gf6a8e84.bin
esptool.py v1.1
Connecting...
Running Cesanta flasher stub...
Flash params set to 0x0020
Writing 528384 @ 0x0... 528384 (100 %)
Wrote 528384 bytes at 0x0 in 12.4 seconds (340.5 kbit/s)...
Leaving...
tony-imac:Downloads tony$
```



```
C:\Users\ladyada\Desktop\pyotp>esptool --port COM95 --baud 115200 write_flash --flash_size=detect 0 adafruit-circuitpython-feather_huzzah-2.1.0.bin
esptool.py v2.2
Connecting...
Detecting chip type... ESP8266
Chip is ESP8266EX
Uploading stub...
Running stub...
Stub running...
Configuring flash size...
Auto-detected Flash size: 4MB
Flash params set to 0x0040
Compressed 595076 bytes to 387307...
Writing at 0x00014000... (25 %)
```

Una vez que la herramienta termina de subir el firmware (normalmente vas a ver una luz azul en el módulo ESP8266 durante el proceso de subida) presiona el botón RESET en la tarjeta ESP8266 o desconéctala y reconéctala a tu computadora. ¡Ya deberías estar listo para usar el firmware de CircuitPython en la tarjeta!

Toma nota que si observas un error de "detect is not a valid flash_size parameter" (o no es capaz de detectar el tamaño del flash) puede que tengas una versión vieja de esptool.py. Para actualizar a la última versión ejecuta el siguiente comando: `pip install --upgrade esptool`

¡Suba librerías y archivos usando Ampy!

La diferencia más grande que vas a ver con las ESP8266 es que necesitas una herramienta especial para manipular archivos. Aprende de `ampy` leyendo esta guía. ¡Es sobre MicroPython, pero para CircuitPython el proceso de instalación y de uso es básicamente idéntico!

Aprende como usar ampy para mover archivos a la ESP8266

<https://adafru.it/q9C>

¡Otras cosas para tomar en cuenta!

- El REPL funciona como se esperaría, así que revisa la página de introducción.
- El almacenamiento de archivos en el chip es compartido con el de CircuitPython, ¡por lo cual si actualizas es posible que pierdas los archivos! Utiliza respaldos.
- Las librerías y API son los mismos para otras tarjetas para CircuitPython.
- Toma nota que la ESP8266 no tiene muchos pines disponibles y solo una entrada analógica en el rango de 0.1-0V. No hay puerto de UART disponible (¡porque es usado por el REPL!).
- No hay salidas analógicas.
- ¡Puedes usar SPI e I2C! Pero necesitas utilizar [bitbangio \(\)](#) para crear el bus de objetos.

Desinstalando CircuitPython

Muchas de nuestras tarjetas se pueden utilizar con múltiples lenguajes de programación. Por ejemplo la Circuit Playground Express puede ser usada con MakeCode, Code.org CS Discoveries, CircuitPython y Arduino.

¿Puede que hayas probado CircuitPython y ahora quieres regresar a MakeCode o Arduino? No es problema.

¡Siempre puedes remover y reinstalar CircuitPython cuando lo desees! ¡Incluso puedes cambiar de parecer todos los días!

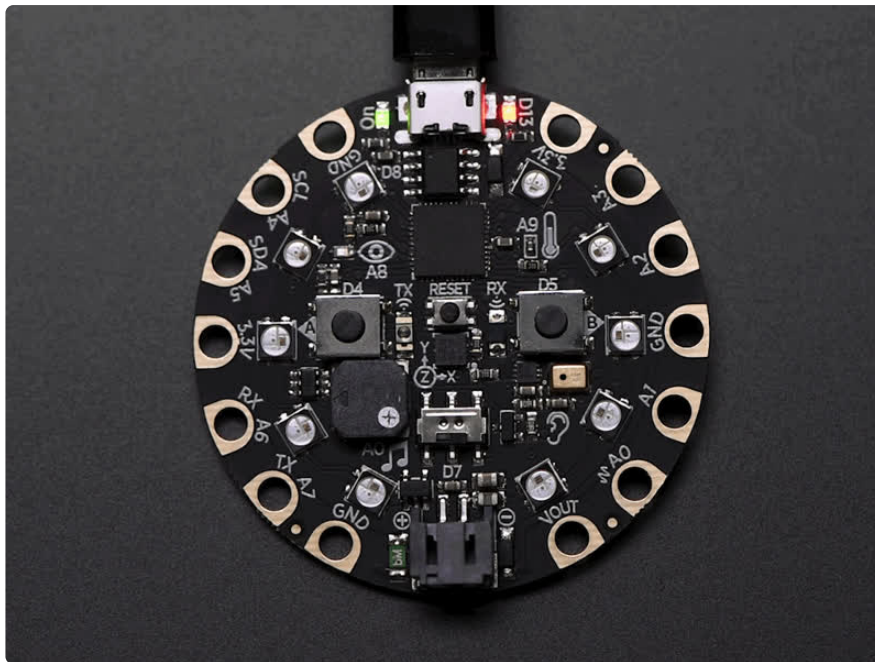
Respalda tu Código

Antes de desinstalar CircuitPython, no te olvides de realizar un respaldo del código que tienes en la pequeña unidad de disco. Eso significa **main.py** o **code.py** y otros archivos, la carpeta **lib**, etc. Es posible que pierdas estos archivos cuando remueves CircuitPython, ¡por lo que los respaldos son clave! Solo arrastra los archivos a una carpeta en tu laptop o computadora de escritorio, como lo harías con cualquier otra memoria USB.

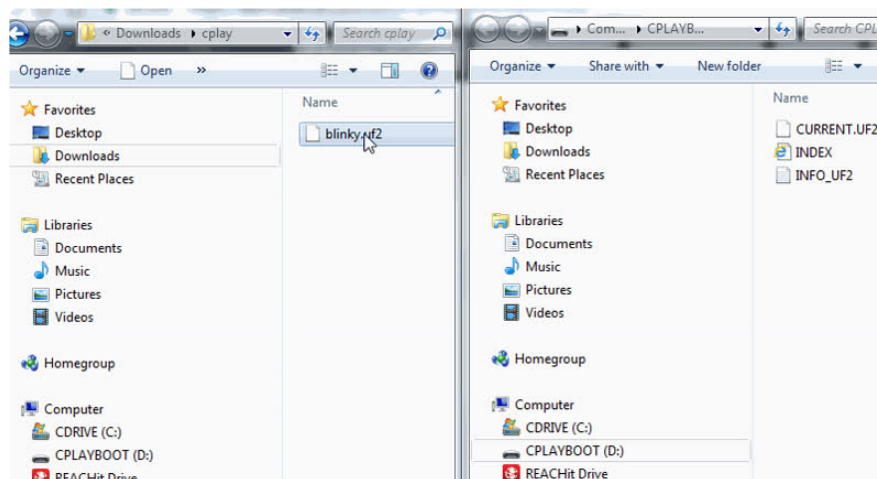
Pasando a la Circuit Playground Express a MakeCode

En la Circuit Playground Express (**por el momento esto NO aplica para la Circuit Playground Bluefruit**), si deseas devolvete a usar MakeCode, es realmente sencillo. Visita makecode.adafruit.com () y encuentra el programa que deseas subir. Da click en Download para descargar el archivo **.uf2** que genera Makecode.

Ahora da doble click en el botón de reset de tu tarjeta CircuitPython hasta que veas las luces LED prender en verde y luego que aparezca el directorio **...BOOT**



Ahora encuentra el archivo **.uf2** descargado de MakeCode y arrástralo a la unidad de disco **...BOOT**



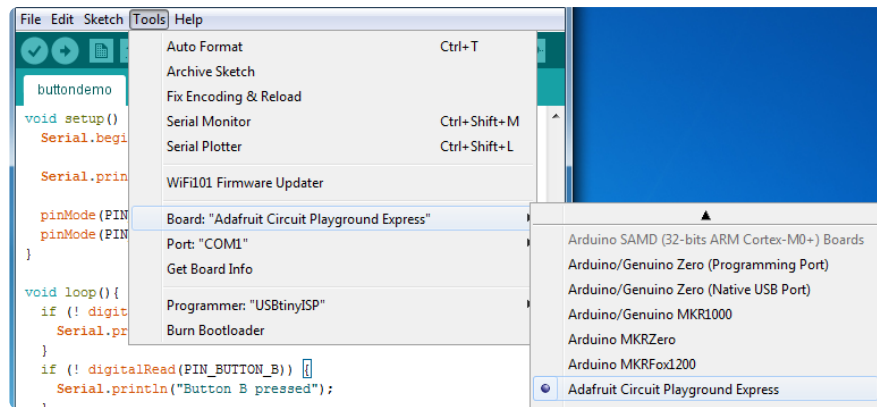
Tu código MakeCode ahora se está ejecutando y CircuitPython ha sido eliminado. De ahora en adelante solo debes dar un **solo click** al botón de reset.

Pasándose a Arduino

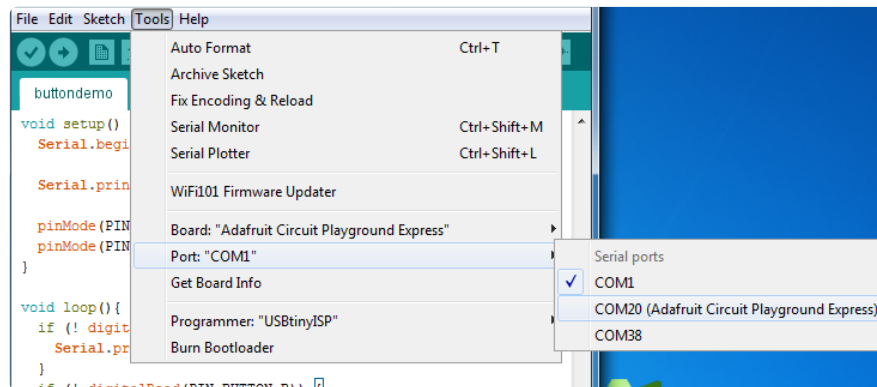
Si quieres mover tu firmware hacia Arduino, es muy sencillo.

Comienza por conectar tu tarjeta y dándole doble click al botón de reset hasta que recibas las luces LED integradas de color verde – igual que con MakeCode.

Dentro del IDE de Arduino, selecciona la tarjeta correcta, diciéndole que use la Circuit Playground Express.



Y seleccionando el puerto correspondiente:



Crear un nuevo programa ejemplo para Parpadear:

```
// la función setup se ejecuta cuando presionas reset o prendes la tarjeta
void setup() {
  // inicialice el pin 13 para salida.
  pinMode(13, OUTPUT);
}

// la función de loop se repite una y otra vez
void loop() {
  digitalWrite(13, HIGH);    // prenda el LED (HIGH o alto, es el nivel de voltaje)
  delay(1000);              // espere un segundo
  digitalWrite(13, LOW);    // apaga el LED pasando el voltaje a LOW (bajo)
  delay(1000);              // espere un segundo
}
```

Asegúrate que los LEDs todavía están de color verde, da click en **Upload** para subir el código de parpadeo. Una vez que ha subido correctamente el puerto serial va a cambiar, **¡así que seleccione el nuevo puerto!**

Una vez que el código de parpadeo ha sido subido, no es necesario volver a dar doble click en el botón de reset para entrar en modo de booteo, Arduino va a resetear la tarjeta de forma automática cuando subes nuevo firmware.

Instalación sin UF2

¡Esta página de instalación es solo necesaria si no tienes un gestor de arranque UF2 instalado (donde sale ...BOOT y le arrastramos archivos)! Esta página es para tarjeta no-Express como Feather M0, Arduino Zero y M0 y otras tarjetas a la medida con chips ATSAMD21.

Subiendo con Bossac – Para las Feather M0 no-Express y Arduino Zero

Nuestras tarjetas viejas de formato Feather con chips M0 no vienen con UF2, sino con un gestor de arranque más sencillo llamado **bossa**. Esto es lo que está instalado en los Arduino Zero y otras tarjetas para CircuitPython que usan chips como ATSAMDx1 o nRF52840. Es el único método de instalación que puedes utilizar si el archivo de instalación de CircuitPython es de tipo **.bin** en lugar de un **.uf2**.

¡Vamos a la línea de comando!

Para subir firmware con bossac requiere utilizar la línea de comando de tu computadora. En Windows, se llama **cmd** o herramienta **powershell**. ¡En mac o Linux, usar **Terminal**!

Descarga el último firmware CircuitPython

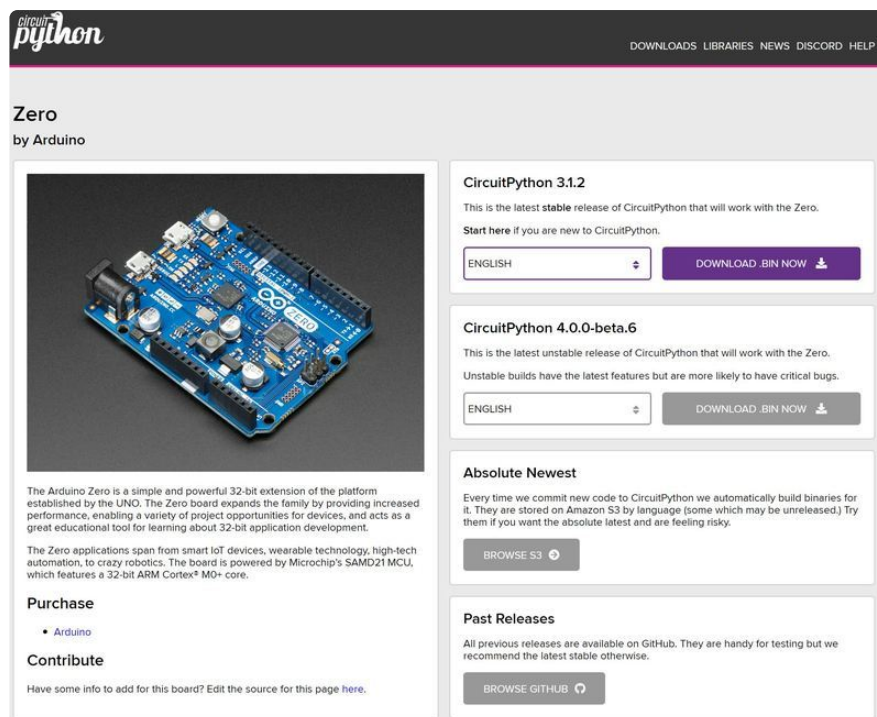
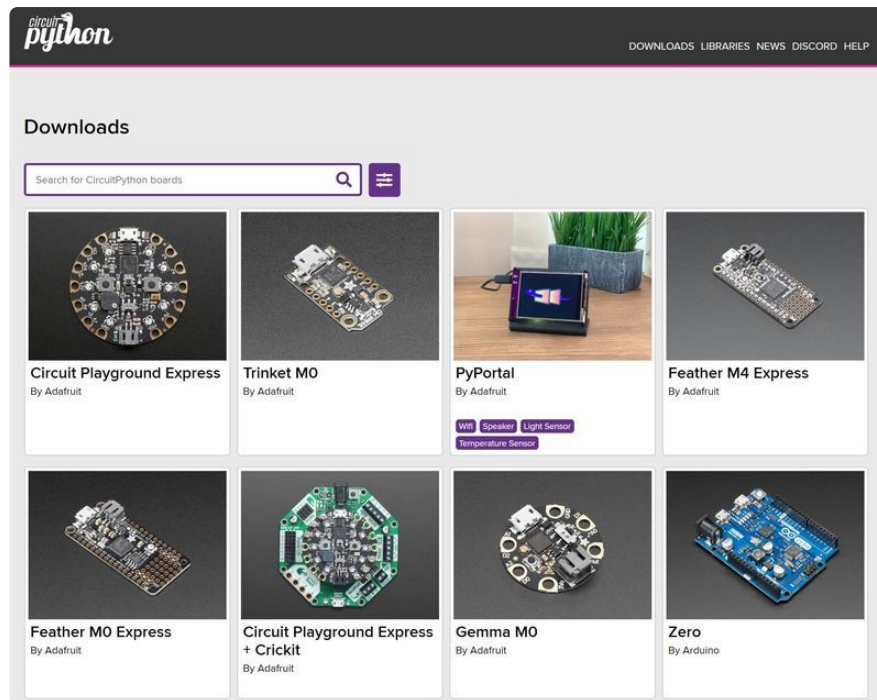
Lo primero que vas a querer hacer es descargar la última versión de CircuitPython.

Desde Abril del 2019, existen dos páginas web para descargar el archivo **.bin** apropiado. Adafruit está migrando sus archivos hacia [CircuitPython.org](https://circuitpython.org) (). Se sugiere que busques el archivo **.bin** compatible para tu tarjeta primero ahí, buscando la tarjeta. Solo tarjetas compatibles que han recibido modificaciones específicas para definir sus pines apropiados van a ser listados en el sitio, por lo cual los desarrolladores deberían enviar sus material (por medio de un pull request) si tienen una tarjeta con CircuitPython la cual desean que salga listada.

Da click al botón verde abajo para buscar tu tarjeta y encontrar el archivo apropiado.

Busque y descargue software para
tu tarjeta desde [CircuitPython.org](https://adafru.it/Em8)

<https://adafru.it/Em8>



De forma alternativa, use GitHub

Por ahora, los archivos de tarjetas para CircuitPython también están disponibles en el repositorio de CircuitPython de Adafruit, aunque a futuro se desea mover a un repositorio separado. Si tienes problemas con CircuitPython.org, trata de descargarlo de GitHub utilizando este segundo botón verde.

Descargue el archivo .bin para tu tarjeta para CircuitPython

<https://adafru.it/v1F>

Una vez descargado, salvar el archivo `.bin` en tu equipo de escritorio, ¡ya casi lo vas a ocupar!

Si estás usando Windows 7, [debes instalar el controlador \(hablamos de ello, en esta página\) \(\)](#) para tener acceso al puerto COM.

Descargando BOSSA

Una vez que tienes el archivo con el firmware, vas a necesitar descargar la herramienta BOSSA, la cual puede cargarle firmware a tarjetas con chips SAMD21/51. Esta herramienta es usada internamente por el IDE de Arduino cuando programa estas tarjetas, sin embargo lo puedes usar para subir tu firmware a la medida.

¡Ten cuidado que **necesitas** una versión **1.7.0 o superior** de bossac para programar tarjetas con chips SAMD21 y SAMD51! Versionas de BOSSA anteriores a 1.7.0 no van a funcionar porque no saben hablar con chips SAMD21/51. **También tener cuidado que versiones 1.9.0 o más nuevas pueden tener un problema de incompatibilidad al cambiar sus parámetros de línea de comando y puede eliminar tu gestor de arranque si no está protegido** (Las tarjetas de Adafruit se envían con gestores de arranque protegidos). Sigue las instrucciones a continuación con mucho cuidado, dependiendo de que versión tengas.

Para subir firmware con `bossac` (la herramienta de línea de comando de BOSSA) primero descarga la última versión de [aquí \(\)](#). La versión `ming32` es para Windows, la versión `apple-darwin` para Mac OSX y varias opciones de `linux` para Linux.

`bossac` solo trabaja con archivos `.bin`, ¡no funciona con archivos `.uf2`!

Prueba bossac

```
tony@mac:bossac-1.7.0$ ./bossac --help
Usage: bossac [OPTION...] [FILE]
Basic Open Source SAM-BA Application (BOSSA) Version 1.7.0
Flash programmer for Atmel SAM devices.
Copyright (c) 2011-2012 Shumatech (http://www.shumatech.com)

Examples:
  bossac -e -w -v -b image.bin # Erase flash, write flash with image.bin,
                                # verify the write, and set boot from flash
  bossac -r0x10000 image.bin   # Read 64KB from flash and store in image.bin

Options:
  -e, --erase           erase the entire flash (keep the 8KB of bootloader for SAM Dxx)
  -w, --write           write FILE to the flash; accelerated when
                        combined with erase option
  -r, --read[=SIZE]    read SIZE from flash and store in FILE;
                        read entire flash if SIZE not specified
  -v, --verify         verify FILE matches flash contents
  -p, --port=PORT      use serial PORT to communicate to device;
                        default behavior is to auto-scan all serial ports
  -b, --boot[=BOOL]    boot from ROM if BOOL is 0;
                        boot from FLASH if BOOL is 1 [default];
                        option is ignored on unsupported devices
  -c, --bod[=BOOL]     no brownout detection if BOOL is 0;
                        brownout detection is on if BOOL is 1 [default]
  -t, --bor[=BOOL]     no brownout reset if BOOL is 0;
                        Brownout reset is on if BOOL is 1 [default]
  -l, --lock[=REGION]  lock the flash REGION as a comma-separated list;
                        lock all if not given [default]
  -u, --unlock[=REGION] unlock the flash REGION as a comma-separated list;
                        unlock all if not given [default]
  -s, --security       set the flash security flag
  -i, --info           display device information
```

Abra una terminal y navegue por la carpeta con la herramienta **bossac**.

Ahora revise que la aplicación funciona, probando con la opción `--help` con **bossac --help**

Si estás utilizando Linux o Mac OSX, vas a necesitar agregar un `./` para decirle que ejecute bossac desde el directorio actual, así **`./bossac --help`**

¡Asegúrate de que observes BOSSA versión 1.7.0 o superior! Y nota la observación abajo sobre la versión 1.9.0 o superior. Si ves una versión anterior es porque descargaste por error una versión anterior y no va a funcionar para subirle a chips SAMD21. Regresa y [descarga la última versión del repositorio de GitHub de BOSSA \(\)](#) como se mencionó arriba.

Selección de puerto para Mac OS

Vas a necesitar saber que puerto usar si estás en Mac.

En la misma terminal, ejecuta el comando **`ls /dev/cu.*`**. Nota los puertos listados, ahora conecta tu tarjeta y ejecuta el comando de nuevo. El dispositivo listado puede llamarse algo similar a **`/dev/cu.usbmodem14301`**. Toma nota del nombre del puerto para usarlo en la sección que sigue sobre **bossac**.

Entra en el gestor de arranque

Vas a tener que darle un empujón a la tarjeta para que entre en modo de gestor de arranque. Lo haces dándole doble-click al botón de reset en la tarjeta. El LED rojo “#13” va a parpadear. Si estás usando un Arduino Zero, asegúrate de que estás conectado al puerto **USB nativo** y no al puerto de depuración/programación.

Una nota importante, si estás utilizando un Arduino MO de Arduino.org, vas a necesitar reemplazar su gestor de arranque con el gestor de arranque de Arduino Zero para que pueda trabajar con BOSSA. Para esto, instala las tarjetas Arduino/Genuino Zero en el IDE de Arduino y luego [sigue estos pasos para subir el gestor de arranque \(\)](#) (usando el puerto de **programación** en la tarjeta). Una vez que has

agregado el gestor de arranque de las Arduino Zero, vas a poder utilizar la M0 con bossac como se describe abajo.

Ejecutando el comando bossac

Con tu tarjeta conectada y corriendo el gestor de arranque, estás listo para subirle el firmware de CircuitPython a la tarjeta. Copia el archivo .bin con el firmware al mismo directorio donde reside la herramienta `bossac`, y luego en una terminal entre en la misma carpeta y ejecute los siguiente comandos, dependiendo de la versión de `bossac` que tengas.

Con bossac versiones 1.9 o superiores debes usar el parámetro `--offset` en la línea de comando, y debe tener el valor correcto para tu tarjeta.

Con bossac versiones 1.9 o superior, le debes indicar en el pámetro `--offset` donde debe comenzar a escribir el firmware en la memoria flash. Este parámetro se agregó desde bossac 1.8.0 por omisión en `0x2000`, pero en 1.9 por omisión se definió a `0x0000`, lo cual no es deseable en la mayoría de casos. Si omites el argumento a bossac 1.9 o superior, probablemente vas a recibir un error de “**Verify Failed**” o verificación fallida. Recuerda cambiar la opción `-p` o `--port` para usar el mismo puerto que en tu Mac.

Reemplaza el nombre abajo por el archivo que has descargado de tipo `.bin`, teniendo en cuenta que ¡varía dependiendo de tu tarjeta!

Utilizando bossac versiones 1.7.0, 1.8

No existe el parámetro `--offset`. Utiliza el comando de la siguiente forma:

```
bossac -p /dev/cu.usbmodem14301 -e -w -v -R adafruit-circuitpython-nombretarjeta-version.bin
```

Por ejemplo,

```
bossac -p /dev/cu.usbmodem14301 -e -w -v -R adafruit-circuitpython-feather_m0_express-3.0.0.bin
```

Usando bossac para versiones 1.9 y superiores

Para tarjetas M0, las cuales tienen 8kB para el gestor de arranque, debes especificar `--offset=0x2000`, por ejemplo:

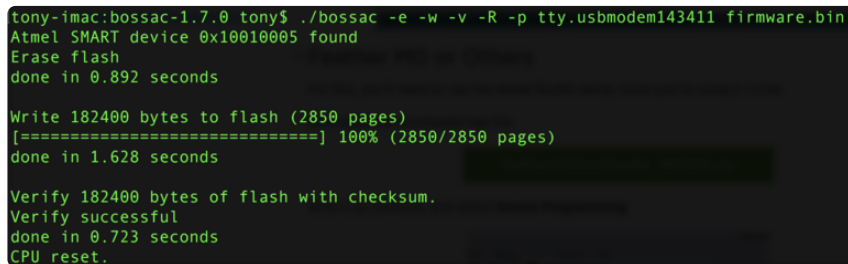
```
bossac -p /dev/cu.usbmodem14301 -e -w -v -R --offset=0x2000 adafruit-circuitpython-feather_m0_express-3.0.0.bin
```

Para tarjetas M4, las cuales tienen 16kB para el gestor de arranque, debes especificar `--offset=0x4000` , por ejemplo:

```
bossac -p /dev/cu.usbmodem14301 -e -w -v -R --offset=0x4000 adafruit-circuitpython-feather_m4_express-3.0.0.bin
```

Esto va a (**e**) borrar la memoria en el chip, (**w**) escribir el archivo indicado, (**v**) verificar la escritura y (**R**) reinicia la tarjeta. En Linux o MacOS vas a querer ejecutar el comando con `sudo ./bossac ...` , o agregarte primero al grupo de **dialout**.

Luego de que BOSSA carga el firmware vas a ver una salida similar a la siguiente:

A terminal window with a dark background and green text. The output shows the execution of the bossac command to flash a firmware file. It reports erasing the flash, writing 182400 bytes to flash (2850 pages) at 100% completion, and verifying the flash with a checksum, all successfully.

```
tony-imac:bossac-1.7.0 tony$ ./bossac -e -w -v -R -p tty.usbmodem143411 firmware.bin
Atmel SMART device 0x10010005 found
Erase flash
done in 0.892 seconds

Write 182400 bytes to flash (2850 pages)
[=====] 100% (2850/2850 pages)
done in 1.628 seconds

Verify 182400 bytes of flash with checksum.
Verify successful
done in 0.723 seconds
CPU reset.
```

Luego de reiniciar, CircuitPython debería quedar corriendo y la unidad de disco **CIRCUITPY** debería aparecer. Puedes reiniciar manualmente la tarjeta dando click al botón de reset, que algunas veces es necesario para despertar la tarjeta. Las tarjetas Express es posible que muestre una advertencia de haber expulsado incorrectamente una unidad de disco USB, la cual se puede ignorar sin problemas. ¡Nada importante se estaba escribiendo a la unidad de disco!

Depuración

De vez en cuando, te vas a topar con problemas mientras trabajas con CircuitPython. Estas son algunas de las cosas que te pueden pasar y como resolverlas.

Mientras continuamos desarrollando CircuitPython y creando nuevas versiones, vamos a dejar de darle mantenimiento a las versiones anteriores. Visita <https://circuitpython.org/downloads> para descargar la última versión de CircuitPython. Favor actualiza CircuitPython y luego visita <https://circuitpython.org/libraries> para descargar la última versión del Agrupado de Librerías.

Siempre Ejecuta la Última Versión de CircuitPython y sus Librerías

Mientras continuamos desarrollando CircuitPython y crear nuevas versiones, vamos a dejar de darle mantenimiento a versiones anteriores. **Usted necesita [actualizarse a la última versión de CircuitPython](https://adafru.it/Em8). (<https://adafru.it/Em8>).**

Usted necesita descargar el Agrupado de Librerías para CircuitPython compatible con tu versión de CircuitPython. Favor actualice CircuitPython y luego [descargue el último agrupado](https://adafru.it/ENC). (<https://adafru.it/ENC>)

Mientras sacamos nuevas versiones de CircuitPython, vamos a dejar de agrupar automáticamente para versiones anteriores en el repositorio de Agrupado de Librerías de CircuitPython de Adafruit. Si desea seguir usando una versión anterior, todavía puede descargar la versión apropiada de `mpy-cross` para una versión en particular de CircuitPython, en el repositorio de CircuitPython y crear sus propios archivos .mpy para librerías. **Sin embargo, es mejor actualizar ambas versiones de CircuitPython y del agrupado de librerías.**

Tengo que seguir usando CircuitPython 3.x o 2.x, ¿donde puedo encontrar librerías compatibles?

Ya no le estamos dando mantenimiento a los agrupados de librerías para CircuitPython 2.x y 3.x. Te recomendamos [actualizar CircuitPython a la última versión](https://adafru.it/Em8) (<https://adafru.it/Em8>) y usar [la versión actuali de las librerías](https://adafru.it/ENC) (<https://adafru.it/ENC>). Sin embargo, si por alguna razón no puedes actualizar, puedes encontrar [la última versión 2.x disponible aquí](https://adafru.it/FJA) (<https://adafru.it/FJA>) y [y la última de 3.x disponible aquí](https://adafru.it/FJB) (<https://adafru.it/FJB>).

Unidades de disco CPLAYBOOT, TRINKETBOOT, FEATHERBOOT, or GEMMABOOT no presentes.

Puede que tenga una tarjeta diferente.

Solo las tarjetas Express de Adafruit y las Trinket M0 y las Gemma M0 vienen con el [gestor de arranque UF2](https://adafru.it/zbX) (<https://adafru.it/zbX>) instalado. Las Feather M0 Basic, Feather M0 Adalogger, y tarjetas similares utilizan un bootloader compatible con Arduino, el cual no tiene una unidad de disco `nombretarjetaBOOT`.

MakeCode

Si estás utilizando un programa en [MakeCode](https://adafru.it/zbY) (<https://adafru.it/zbY>) presiona el botón de reset solo una vez para que aparezca la unidad de disco **CPLAYBOOT**. Presionándola dos veces, no va a funcionar.

Windows 10

¿Instalaste el paquete con Controladoras para Windows de Adafruit por error? No necesitas instalar este paquete en Windows 10 para la mayoría de las tarjetas. La versión anterior (v1.5) puede causar problemas reconociendo tu dispositivo. Vaya a **Settings -> Apps** y desinstale controladoras de "Adafruit".

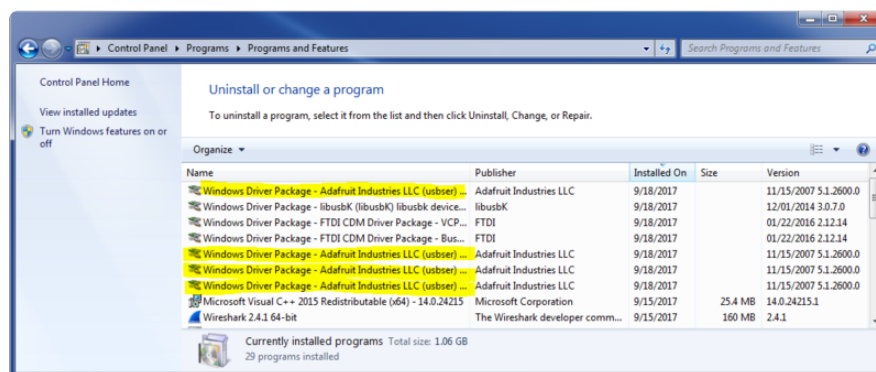
Windows 7

La versión 2.0.0.0 o posterior de las controladoras para Windows de Adafruit van a arreglar el problema de que no aparezca la unidad de disco **nombretarjetaBOOT**. Para resolver esto, primero desinstale las versiones anteriores de las controladoras:

- Desconecte cualquier tarjeta. En **Uninstall or Change a Program (Control Panel->Programs->Uninstall a program)**, desinstale cualquier cosa de nombre "Windows Driver Package - Adafruit Industries LLC ...".

MacOS

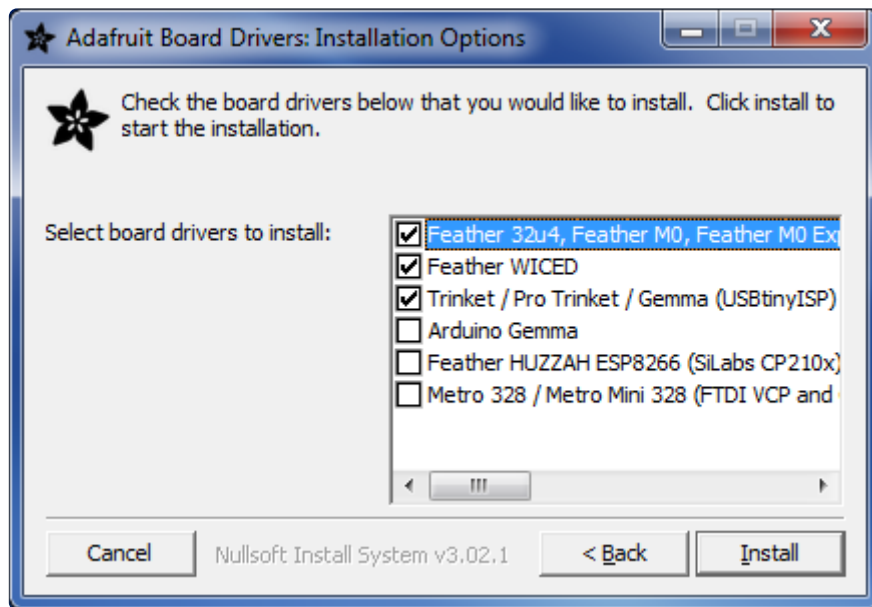
DriveDX y su controladora acompañante **SAT SMART Driver** pueden interferir con ver una unidad de disco BOOT. [Observa este mensaje del foro](https://adafru.it/sTc) (<https://adafru.it/sTc>) para saber como arreglar este problema.



- Ahora instale el nuevo paquete de Controladoras de Adafruit para Windows, versión 2.3.0.0 (o superior):

<https://adafru.it/AB0>

- Cuando ejecutas el instalador, vas a observar una lista de controladoras a escoger. Puedes marcar y desmarcar las cajas para escoger los controladores a instalar.



¡Ya estás listo! Realice una prueba desconectando y reconectando la tarjeta. Deberías ver la unidad de disco **CIRCUITPY**, y cuando le das doble click al botón de reset (o un solo click en una Circuit Playground Express corriendo MakeCode), deberías ver la unidad de disco apropiada tipo **nombretarjetaB00T**.

¡Déjenos saber en los [foros de servicio técnico de Adafruit](https://adafru.it/jlf) (<https://adafru.it/jlf>) o en el [canal de Adafruit en Discord](#) () si esto no te funciona!

Windows Explorer se queda pegado al acceder la unidad de disco **nombretarjetaB00T**

En Windows, algunas aplicaciones de tercero pueden causar problemas. El síntoma es que tratas de acceder la unidad de disco **nombretarjetaB00T**, y Windows o el Explorador de Windows parece que se queda pegado. Estas aplicaciones son causantes conocidos:

- **AIDA64**: para arreglarlo, detenga la aplicación. El problema ha sido reportado a AIDA64. Ellos consiguieron hardware para probar y sacaron una versión beta que resuelve este problema. Esto puede que se haya integrado con la última versión. Déjanos saber en los foros si lo pruebas.
- **Hard Disk Sentinel**

- **Anti-virus Kaspersky:** Para arreglarlo, es posible que tengas que deshabilitar Kaspersky del todo. Deshabilitar algunos aspectos de Kaspersky puede que no arregle el problema. El problema ha sido reportado a Kaspersky.

Copiar un UF2 a la unidad de disco **nombretarjetaB00T** se queda pegado con un 0% de copia completada

En Windows, el **utilitario de Western Digital (WD)** que se incluye con las unidades de disco externas USB, interfiere con la copia de archivos UF2 hacia la unidad de disco **nombretarjetaB00T**. Desinstala el utilitario para arreglar el problema.

La Unidad de Disco CIRCUITPY no Aparece

El **anti-virus Kaspersky** puede bloquear el que aparezca la unidad de disco **CIRCUITPY**. Aún no hemos encontrado un cambio de configuración que evite que suceda. Una desinstalación completa de Kaspersky arregla el problema.

El **anti-virus Norton** puede interferir con **CIRCUITPY**. Un usuario ha reportado este problema en Windows 7. El usuario apagó tanto la Pared de Fuego Inteligente (o Smart Firewall) y Auto Protect, y **CIRCUITPY** entonces apareció.

La Consola Serial de Mu no Imprime Nada

Pasa a veces que la consola serial no va a imprimir nada, por ejemplo, cuando no hay código corriendo en ese momento, o cuando un código sin salida serial ya se está ejecutando antes de abrir la consola. Sin embargo, si te encuentras en una situación donde consideras que debería estar imprimiendo algo como por ejemplo un mensaje de error, considera lo siguiente.

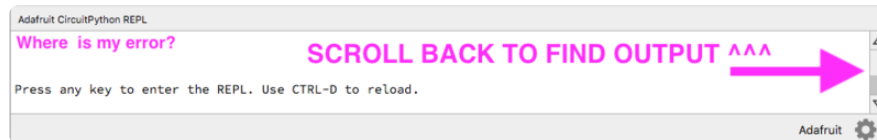
Dependiendo del tamaño de tu pantalla o de la ventana de Mu, cuando abres la consola serial, el panel de la consola serial puede que quede muy pequeño. Esto puede ser incómodo. ¡Un mensaje de error típico en CircuitPython toma 10 líneas para desplegarse!

```
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 7  
SyntaxError: invalid syntax
```

Press any key to enter the REPL. Use CTRL-D to reload.

¡Errores más complejos necesitan todavía más líneas!

Por lo tanto, si tu panel de consola serial es de menos de 5 líneas, puede que solo veas líneas en blanco, o líneas en blanco seguidas de **Press any key to enter the REPL. Use CTRL-D to reload.** Si este es el caso, necesitas pararte con el mouse encima de la parte superior del panel para poderle cambiar su tamaño o utilizar la barra de desplazamiento al lado derecho para encontrar tu mensaje.



Esto aplica para cualquier tipo de salida serial, ya sea un mensaje de error o un mensaje de un print. Así que antes de que trates de depurar tu problema del lado del hardware, primero asegúrate de que no hay mensajes que puedas no estar viendo a causa del tamaño del panel de consola serial.

Luz de Estado RGB para CircuitPython

Casi todas las tarjetas para CircuitPython de Adafruit, tienen en la tarjeta un solo LED RGB de tipo NeoPixel o DotStar, que indica el estado de CircuitPython. Algunas tarjetas que se diseñaron antes de que existiera CircuitPython, tales como las Feather M0 Basic, no lo tienen.

Las Circuit Playground Express y las Circuit Playground Bluefruit tienen múltiples LEDs RGB, pero NO un LED de estado. Los LEDs prenden en verde cuando está dentro del modo de gestor de arranque. Ellos NO indican el estado mientras CircuitPython se ejecuta.

Esto es lo que los colores y su parpadeo significan:

- **VERDE** fijo: se está ejecutando `code.py` (o `code.txt`, o `main.py`, o `main.txt`)
- **VERDE** parpadeando: `code.py` (etc.) ha terminado de correr o no existe
- **AMARILLO** fijo al inicio: (4.0.0-alpha.5 o más reciente) CircuitPython está esperando un reset para indicar que debería entrar en modo a prueba de fallos (o safe mode)
- **AMARILLO** parpadeando: CircuitPython está en modo a prueba de fallos: tuvo una caída y se reinició
- **BLANCO** fijo: El REPL está corriendo
- **AZUL** fijo: `boot.py` está corriendo

El parpadeo de múltiples colores en secuencia indica una excepción de Python donde se indica el número de línea del error. El color del primer parpadeo indica el tipo de error:

- **VERDE**: IndentationError
- **CIAN**: SyntaxError
- **BLANCO**: NameError
- **NARANJA**: OSError
- **MORADO**: ValueError
- **AMARILLO**: other error

Luego sigue la secuencia de parpadeo que indica el número de línea, incluyendo un valor por posición. Parpadeo **BLANCO** son miles, **AZUL** las centenas, **AMARILLO** las decenas, y **CIAN** las unidades. Así que por ejemplo, un error en la línea 32 parpadearía en **AMARILLO** tres veces y en **CIAN** dos veces más. Los ceros se representan por apagados de larga duración.

ValueError: Incompatible **.mpy** file.

Esto sucede cuando se importa un módulo que ha sido almacenado como binario **.mpy**, el cual ha sido creado con una versión diferente de CircuitPython a la que estás usando para cargarlo. En particular, el formato binario **.mpy** cambió entre las versiones 2.x y 3.x de CircuitPython, así como también entre 1.x y 2.x.

Así que por ejemplo, si has actualizado a CircuitPython 3.x desde 2.x vas a necesitar descargar la última versión de la librería que inició el problema cuando se ejecutó el import. Todas están disponibles en el [Agrupado de Librerías de Adafruit \(https://adafru.it/y8E\)](https://adafru.it/y8E).

Asegúrate de descargar una versión con 2.0.0 o superior en el nombre de archivo si estás usando CircuitPython 2.2.4, y la versión 3.0.0 o superior si estás utilizando CircuitPython 3.0.

Problemas con la Unidad de Disco CIRCUITPY

Puede que te pase que ya no puedes salvar archivos en la unidad de disco **CIRCUITPY**. Puede que te pase que **CIRCUITPY** ya no sale en tu explorador de archivos, o ya aparece como **NO_NAME**. Estos son indicadores de que tu sistema de archivos tiene problemas.

Primero revisa - ¿Has utilizado Arduino para programar tu tarjeta? Si es así, CircuitPython ya no puede dar los servicios de USB. Reinicie la tarjeta para recibir la

unidad de disco `nombretarjetaBOOT` en lugar de la unidad de disco `CIRCUITPY`, copie la última versión de `CIRCUITPYTHON` (`.uf2`) hacia la tarjeta y reinicie. Ya debería aparecer `CIRCUITPY`.

Si todavía sigue mal - Es posible que el sistema de archivos esté corrupto como resultado de no expulsar de forma segura a la unidad de disco `CIRCUITPY` antes de presionar el botón de reset o de desconectarla del USB. Esto puede pasar en Windows, Mac o Linux.

En este caso, la tarjeta debe ser borrada completamente y CircuitPython recargado a la tarjeta.

VAS A PERDER todo en la tarjeta cuando completes los siguientes pasos. Si es posible, realiza una copia de tu código antes de continuar.

Forma sencilla: Usando `storage.erase_filesystem()`

Desde la versión 2.3.0, CircuitPython incluye una función para borrar y dar formato al sistema de archivos. Si tienes una versión anterior de CircuitPython en tu tarjeta, puedes [actualiza a la última versión \(https://adafru.it/Amd\)](https://adafru.it/Amd) para poder hacer esto.

1. [Conéctese al REPL \(https://adafru.it/Bec\)](https://adafru.it/Bec) usando Mu or la aplicación de terminal.
2. Escriba:

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

CIRCUITPY se va a borrar y a dar formato, y luego tu tarjeta se reiniciará. ¡Listo!

Forma antigua: Para las Circuit Playground Express, Feather M0 Express, y Metro M0 Express:

Si no puedes entrar en el REPL, o estás corriendo una versión anterior a CircuitPython 2.3.0 y no deseas actualizar, puedes hacer esto.

1. Descargue el archivo de borrado apropiado para tu tarjeta:

Circuit Playground Express

<https://adafru.it/Adl>

Feather M0 Express

<https://adafru.it/AdJ>

Feather M4 Express

<https://adafru.it/EVK>

Metro M0 Express

<https://adafru.it/AdK>

Metro M4 Express QSPI Eraser

<https://adafru.it/EoM>

Trellis M4 Express (QSPI)

<https://adafru.it/DjD>

Grand Central M4 Express (QSPI)

<https://adafru.it/DBA>

PyPortal M4 Express (QSPI)

<https://adafru.it/Eca>

Circuit Playground Bluefruit (QSPI)

<https://adafru.it/Gnc>

Monster M4SK (QSPI)

<https://adafru.it/GAN>

PyBadge/PyGamer QSPI Eraser.UF2

<https://adafru.it/GAO>

CLUE_Flash_Erase.UF2

<https://adafru.it/Jat>

2. Realice doble click en el botón de reset de la tarjeta para que aparezca la unidad de disco **nombretarjetaB00T**.

3. Arrastre el archivo de borrado **.uf2** a la unidad de disco **nombretarjetaB00T**.

4. El LED de estado se va a prender de color amarillo o azul, indicando que el borrado a comenzado.

5. Luego de aproximadamente 15 segundos, el LED de estado se prenderá de color verde. En las NeoTrellis M4 este es el primer NeoPixel de la rejilla.

6. Realice doble click en el botón de reset de la tarjeta para que aparezca la unidad de disco `nombratarjetaBOOT`.

7. [Arrastre la versión correcta de CircuitPython en archivo \(https://adafru.it/Amd\)](https://adafru.it/Amd) `.uf2` hacia la unidad de disco `nombratarjetaBOOT`.

Debería reiniciar automáticamente y deberías ver de nuevo en tu explorador de archivos a la unidad de disco `CIRCUITPY`,

Si el LED parpadeo en rojo durante el paso 5, significa que el borrado falló. Repita los pasos desde el 2.

[Si aún no lo ha hecho, descargue la última versión de CircuitPython para tu tarjeta, en la página de instalación](https://adafru.it/Amd)
[If you haven't already downloaded the latest release of CircuitPython for your board, check out the installation page \(https://adafru.it/Amd\).](https://adafru.it/Amd)

¡También vas a ocupar instalar las librerías y tu código!

Forma antigua: Para tarjetas no-Express con gestor de arranque UF2 (Gemma M0, Trinket M0):

Si no puedes recibir el REPL, o estás ejecutando una versión anterior a CircuitPython 2.3.0 y no quieres actualizar, puedes hacer esto.

1. Descargue el archivo de borrado:

Gemma M0, Trinket M0

<https://adafru.it/AdL>

2. Realice doble click en el botón de reset de la tarjeta para que aparezca la unidad de disco `nombratarjetaBOOT`.

3. Arrastre el archivo de borrado `.uf2` hacia la unidad de disco `nombratarjetaBOOT`.

4. El LED de estado va a parpadear, y la unidad `nombratarjetaBOOT` aparece.

5. [Arrastre una versión reciente de CircuitPython para su tarjeta \(https://adafru.it/Amd\)](https://adafru.it/Amd) en archivo `.uf2` hacia la unidad de disco `nombratarjetaBOOT`.

Debería reiniciarse y verse la unidad de disco `CIRCUITPY` de nuevo en el explorador de archivos.

Si no lo has hecho, descarga la última versión de CircuitPython para tu tarjeta en la [página de instalación \(https://adafru.it/Amd\)](https://adafru.it/Amd) ¡También vas a ocupar instalar las librerías y tu código!

Forma antigua: Para tarjetas no-Express sin gestor de arranque UF2 (Feather M0 Basic Proto, Feather Adalogger, Arduino Zero):

Si estás corriendo un versión anterior a CircuitPython 2.3.0 y no quieres actualizar o no puedes recibir al REPL, puedes hacer esto.

[Sigue las instrucciones para cargar CircuitPython usando bossac \(https://adafru.it/Bed\)](https://adafru.it/Bed), con la cual se va a borrar y a dar formato a **CIRCUITPY**.

Quedándose Sin Espacio en Tarjetas No-Express

El sistema de archivos de la tarjeta es muy pequeño. (Más pequeños que los viejos discos floppy) Así que es posible que te vayas a quedar sin espacio, pero ¡sin pánico! Hay un par de formas para liberar espacio.

La tarjeta se envía con un controlador serial para Windows 7. Sin problemas puedes eliminar el archivo si no lo necesitas o si ya lo tienes instalado. Son como 12KiB.

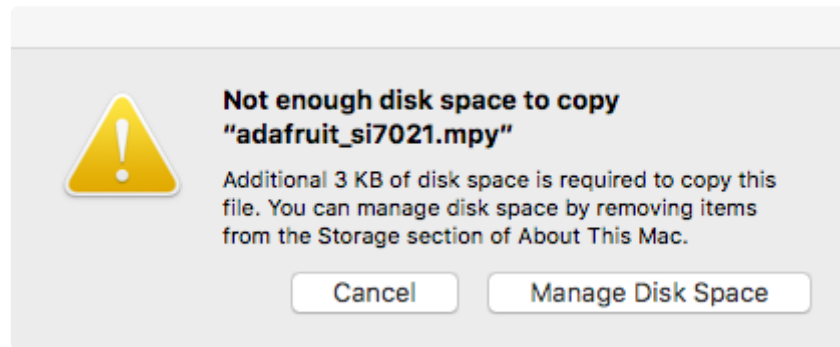
¡Borra algo!

La forma más sencilla de liberar espacio es borrando archivos de la unidad de disco. Tal vez hay librerías en la carpeta de **lib** que ya no estás utilizando o código de prueba que no se usa. No borre completamente la carpeta **lib**, solamente borre lo que no haga falta.

Use tabs

Una característica única de Python es que la indentación del código importa. Usualmente la recomendación es indentar el código con cuatro espacio por sangría. En general, también recomendamos eso. **Sin embargo**, un truco para almacenar más código legible por humanos es utilizar un solo tab para la sangría. De esta forma se usa 1/4 del espacio de almacenamiento para indentar, y esto puede ser significativo cuando contamos en bytes.

A MacOS le encantan los archivos extra.



Por suerte, puedes deshabilitar algunos de los archivos escondidos adicionales que MacOS agregar, corriendo un par de comandos para deshabilitar el indexado y crear archivos de cero bytes para reservar el lugar. Sigue los pasos a continuación para maximizar la cantidad de espacio disponible en MacOS:

Prevenir y Remover Archivos Escondidos de MacOS Hidden

Primero, encuentre el nombre del volumen de disco para tu tarjeta. Con la tarjeta conectada, ejecutas ese comando en una terminal para listas las unidades de disco:

```
ls -l /Volumes
```

Buscar una unidad de disco con un nombre como **CIRCUITPY** (el nombre por omisión en CircuitPython). La ruta completa del volumen es la ruta **/Volumes/CIRCUITPY**.

Ahora siga los [pasos descritos en esta pregunta \(https://adafru.it/u1c\)](https://adafru.it/u1c) donde ejecutamos estos comandos para detender que los archivos escondidos sean creados en la tarjeta:

```
mdutil -i off /Volumes/CIRCUITPY
cd /Volumes/CIRCUITPY
rm -rf .{,._}{fseventsd,Spotlight-V*,Trashes}
mkdir .fseventsd
touch .fseventsd/no_log .metadata_never_index .Trashes
cd -
```

Reemplace **/Volumes/CIRCUITPY** en los comandos descritos arriba, si el nombre de tu tarjeta es diferente. En este paso, los archivos escondidos han sido borrados de tu tarjeta y se va a prevenir que algunos archivos escondidos sean creados.

De forma alternativa, desde CircuitPython 4.x y superiores, los archivos especiales y carpetas descritos van a ser creados automáticamente luego de borrar y dar formato al sistema de archivos. **Advertencia: ¡Primero salve sus archivos!** Haga esto en el REPL:

```
>>> import storage
>>> storage.erase_filesystem
```

Sin embargo, todavía hay casos donde archivos escondidos pueden ser creados por MacOS. En particular, si copias un archivo que ha sido descargado de internet, va a tener metadata que MacOS almacena como un archivo escondido. Por suerte, puedes correr un comando de copia desde una terminal para copiar archivos **sin** este archivo escondido de metadatos. Observe los siguientes pasos.

Copie Archivos en MacOS Sin Crear Archivos Escondidos

Una vez que has deshabilitado y removido archivos escondidos con los comandos descritos arriba en MacOS, vas a necesitar realizar una copia cuidadosa de archivos para prevenir que futuros archivos escondidos sean creados. Lastimosamente, **no se puede** arrastrar archivos desde el Finder, dado que va a crear los archivos escondidos de metadatos en algunos casos (para archivos descargados de internet, como módulos de Adafruit).

Para copiar un archivo o carpeta, utilice la opción **-X** del comando **cp** en una terminal. Por ejemplo, para copiar el archivo **foo.mpy** a la tarjeta, use un comando similar a:

```
cp -X foo.mpy /Volumes/CIRCUITPY
```

(Reemplace **foo.mpy** con el nombre del archivo que quiere copiar) O para copiar una carpeta con todos sus archivos y carpetas que contiene, use un comando similar a:

```
cp -rX folder_to_copy /Volumes/CIRCUITPY
```

Si estás copiando hacia la carpeta **lib**, o hacia otra carpeta, es bueno verificar que exista, antes de copiarle algo.

```
# ¡si lib no existe, vas a crear un archivo llamado lib !
cp -X foo.mpy /Volumes/CIRCUITPY/lib
# Esta forma es preferida, y se va a quejar si la carpeta lib
# no existe.
cp -X foo.mpy /Volumes/CIRCUITPY/lib/
```

Otros tips para Ahorro de Espacio en MacOS

Si desea ver la cantidad de espacio utilizada en la unidad de disco, y manualmente borrar archivos escondidos, esta es la forma. Primero, liste la cantidad de espacio utilizada en la unidad de disco **CIRCUITPY** con el comando **df**:

```
(venv) tannewt@shallan:/Volumes $ df -h /Volumes/CIRCUITPY/
Filesystem      Size  Used Avail Capacity iused ifree %used  Mounted on
/dev/disk3s1    59Ki  54Ki  5.5Ki   91%    128     0  100%  /Volumes/CIRCUITPY

(venv) tannewt@shallan:/Volumes $ ls -a CIRCUITPY/
./
../
.Trashes*
.fsevents/
README.txt*
Windows 7 Driver/
boot_out.txt*
code.py*
lib/
original_code.py*
```

Removamos los archivos `./` primero.

```
(venv) tannewt@shallan:/Volumes $ df -h /Volumes/CIRCUITPY/
Filesystem      Size  Used Avail Capacity iused ifree %used  Mounted on
/dev/disk3s1    59Ki  54Ki  5.5Ki   91%    128     0  100%  /Volumes/CIRCUITPY

(venv) tannewt@shallan:/Volumes $ ls -a CIRCUITPY/
./
../
.Trashes*
.fsevents/
README.txt*
Windows 7 Driver/
boot_out.txt*
code.py*
lib/
original_code.py*

(venv) tannewt@shallan:/Volumes $ rm CIRCUITPY/.*
(venv) tannewt@shallan:/Volumes $ df -h /Volumes/CIRCUITPY/
Filesystem      Size  Used Avail Capacity iused ifree %used  Mounted on
/dev/disk3s1    59Ki  42Ki  18Ki   71%    128     0  100%  /Volumes/CIRCUITPY

(venv) tannewt@shallan:/Volumes $ ls -a CIRCUITPY/
./
../
.Trashes/
.fsevents/
Windows 7 Driver/
lib/
boot_out.txt*
original_code.py*
README.txt*
code.py*
```

¡Hey! ¡Tenemos 13Ki más que antes! ¡Este espacio ahora puede ser usado para librerías y código!

Esenciales para CircuitPython



Has repasado toda la guía de Bienvenido a CircuitPython. Ya tienes todo preparado y estás corriendo CircuitPython. ¡Excelente! ¿Y ahora que? ¡Los esenciales para CircuitPython!

Hay varios módulos núcleo integrados en CircuitPython y tienes disponible librerías populares. La guía de Esenciales te introducirá a ellas, y te va a dar un ejemplo de como usar cada una.

¡Es hora de comenzar a aprender con los [Esenciales para CircuitPython \(https://adafruit.it/cpy-essentials\)](https://adafruit.it/cpy-essentials)!