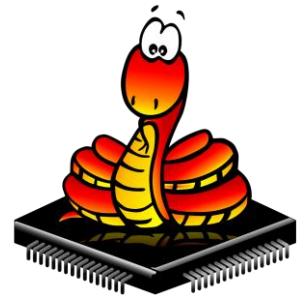


# PyCon Latam

2024



## “Micropython”

### Firts Contact



# Contenido

## Contenido

<b>Primer contacto con Micropython .....</b>	3
<b>Sistemas embebidos .....</b>	3
<b>¿Qué es Micropython?.....</b>	3
<b>¿Por qué usar Micropython?.....</b>	4
<b>¿Qué es un Microcontrolador?.....</b>	4
<b>Comunicación.....</b>	5
<b>Pines GPIO .....</b>	5
<b>Programación .....</b>	5
<b>La tarjeta Arduino Nano RP2040 .....</b>	5
<b>Instalación del firmware MicroPython .....</b>	6
<b>Obtener el IDE Arduino Lab for MicroPython .....</b>	9
<b>Primer programa en MicroPython.....</b>	11
<b>Pinout Arduino Nano RP2040.....</b>	14
<b>Pinout Microcontrolador RP2040 .....</b>	14
<b>Mostrar todos los pines disponibles .....</b>	15
<b>Accediendo a sensores con MicroPython.....</b>	16
<b>IoT .....</b>	18
<b>Conectividad WiFi.....</b>	18
<b>A) Búsqueda de redes wifi .....</b>	18
<b>B) Hacer una solicitud a una API.....</b>	19
<b>Conclusiones .....</b>	20

## Primer contacto con Micropython

El objetivo de esta platica es ofrecer una pequeña muestra sobre cómo implementar Micropython usando la tarjeta **Arduino Nano RP2040 Connect**.

### Sistemas embebidos

Un sistema embebido es un sistema informático que se compone de hardware y software, y que se diseña para realizar una función específica.

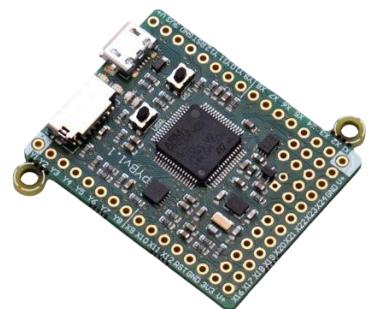
Aunque los sistemas embebidos existen desde hace varias décadas, sólo en tiempos recientes han evolucionado hacia tecnologías más fáciles de usar para aficionados sin conocimientos especializados. Con la llegada de nuevos diseños de microcontroladores y lenguajes de programación más accesibles, los principiantes pueden empezar a utilizar estos complejos sistemas de forma más eficiente.



La combinación de **Arduino** y **MicroPython** es un gran ejemplo: por un lado, hay una plataforma de hardware abierto (open source hardware) bastante difundida, con una comunidad compuesta de millones de personas distribuidas por todo el mundo, con una gran cantidad de documentación disponible, lo que la hace muy accesible y sencilla de usar, mientras que, por el otro, un conocido lenguaje de programación que ha sido adaptado para trabajar en microcontroladores.

### ¿Qué es Micropython?

MicroPython es una implementación eficiente (reducida) del lenguaje de programación Python 3 diseñada para ejecutarse en microcontroladores, creada por el físico y programador australiano **Damien George** en 2014 gracias a una exitosa campaña en Kickstarter, junto a la **Pyboard**, la placa oficial de Micropython, desde entonces se ha portado a diferentes plataformas.



MicroPython ofrece la simplicidad de Python combinada con el control de bajo nivel necesario para el desarrollo de sistemas embebidos, lo que hace que la programación de hardware sea más accesible, incluso para aquellos con experiencia mínima en microcontroladores y electrónica.

## ¿Por qué usar Micropython?

Una de las necesidades (o moda) que se ha incrementado en los últimos años es el IoT, si, el famoso Internet de las cosas, que la podemos definir como:

*El Internet de las cosas (IoT) es el proceso que permite conectar los elementos físicos cotidianos al Internet: desde los objetos domésticos comunes, como los focos (bombillas de luz), hasta los recursos para la atención de la salud, como los dispositivos médicos; las prendas (wearables) y los accesorios personales inteligentes; e incluso los sistemas de las ciudades inteligentes.*

Fuente: <https://www.redhat.com/es/topics/internet-of-things/what-is-iot>

MicroPython es una de las opciones preferidas para su uso en el IoT por varias razones. La primera es que **MicroPython está basado en Python**, lo que lo hace accesible para muchos desarrolladores, independientemente de su experiencia. Esto reduce significativamente la curva de aprendizaje para quienes son nuevos en los sistemas embebidos.

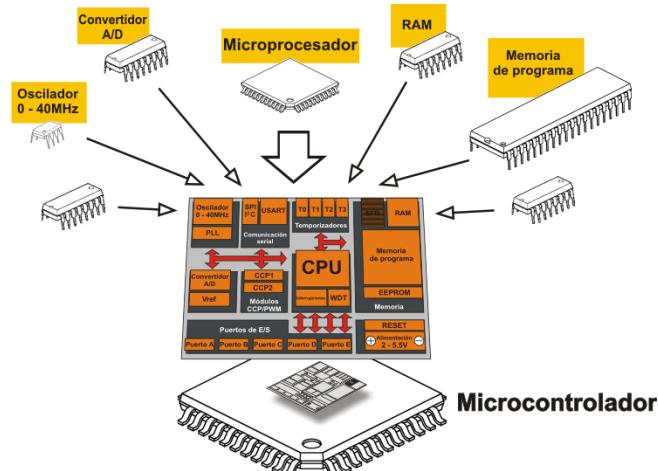
En segundo lugar, el tamaño reducido y la eficiencia de MicroPython son una buena opción para microcontroladores con recursos limitados, ya que permiten que los dispositivos IoT funcionen con recursos de hardware mínimos. Esto es muy relevante para implementaciones de IoT rentables y energéticamente eficientes.

Finalmente, la comunidad activa de MicroPython facilita la resolución de muchos problemas que pueden surgir durante el desarrollo del proyecto. Los desarrolladores pueden aprovechar muchos códigos y módulos existentes que admiten diferentes funcionalidades y protocolos, como **MQTT**.

En resumen, MicroPython es una opción ideal para proyectos de IoT debido a su simplicidad, eficiencia de recursos y soporte comunitario.

## ¿Qué es un Microcontrolador?

Un microcontrolador es un circuito integrado que en su interior contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida y periféricos. Estas partes están interconectadas dentro del microcontrolador, y en conjunto forman lo que se le conoce como microcomputadora. Se puede decir con toda propiedad que un microcontrolador es una microcomputadora completa encapsulada en un circuito integrado.



## Comunicación

### Pines GPIO

Se denomina por GPIO (General Purpose Input/Output) a los pines presentes en diversos circuitos integrados (como los microcontroladores, microprocesadores, SoCs, FPGAs, etc.) que son de propósito general, esto es, que no tienen una funcionalidad específica pre-fijada de fábrica (en un principio no realizan ninguna función), sino que están disponibles para ser configurados y usados por el usuario/desarrollador para que cumpla con la finalidad que se requiera.

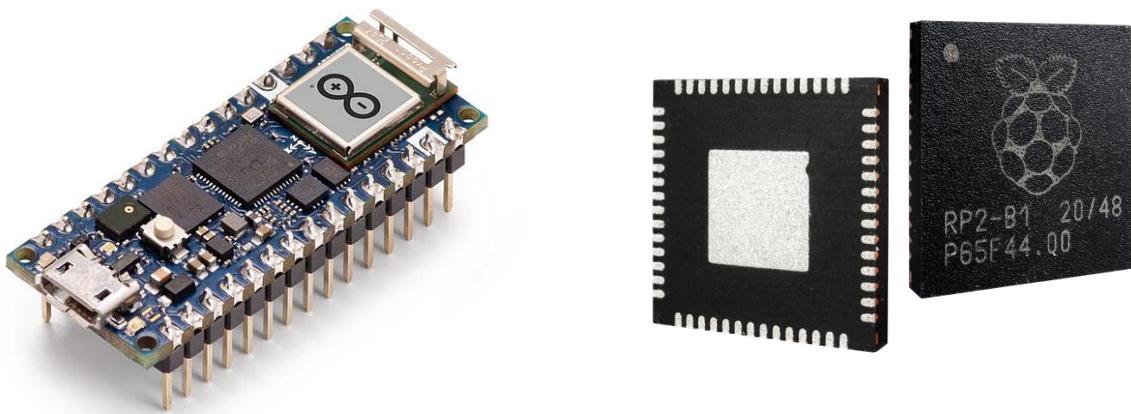
## Programación

Hoy en día, para programar un microcontrolador, se cuenta con diferentes lenguajes, entre ellos: Ensamblador, C, C++, Rust, Micropython y hasta en JS 😊.

### La tarjeta Arduino Nano RP2040

Arduino Nano RP2040 Connect es una tarjeta electrónica desarrollada por Arduino, está basada en el microcontrolador RP2040, creado por la fundación Raspberry Pi. Este microcontrolador contiene un ARM Cortex-M0+ de doble núcleo con una gran cantidad de puertos (pines) GPIO (General Purpose Input/Output, Entrada/Salida de Propósito General) y soporte para varias interfaces de comunicación.

Contiene un módulo WiFi (ESP32) de la compañía u-blox, lo que la hace adecuada para diseños de IoT. La conexión WiFi permite utilizar varios protocolos de comunicación, como por ejemplo el conocido MQTT.



Esta tarjeta fue diseñada para ser compacta, versátil y fácil de usar, ya que incluye todo el conocido ecosistema Arduino, lo que permite a cualquier persona programarla con el IDE clásico de Arduino usando C++ o el nuevo **Arduino Lab for MicroPython**, después de un procedimiento de configuración muy sencillo, que se explica a continuación.

## Instalación del firmware MicroPython

Instalar el firmware MicroPython en la tarjeta es el primer paso para realizar.

La tarjeta Arduino Nano RP2040 Connect no soporta de forma nativa el lenguaje MicroPython, por lo que este paso es necesario.

Puede descargar el firmware correcto desde aquí: <https://docs.arduino.cc/micropython/>

Existen varias versiones, se recomienda usar la última versión más estable, a la fecha de elaboración de este manual es la: (stable) 20240222-v1.22.2.uf2

### Firmware

Arduino has developed MicroPython support for the boards listed below. Here you can find the latest firmwares, as well as preview versions and older releases.

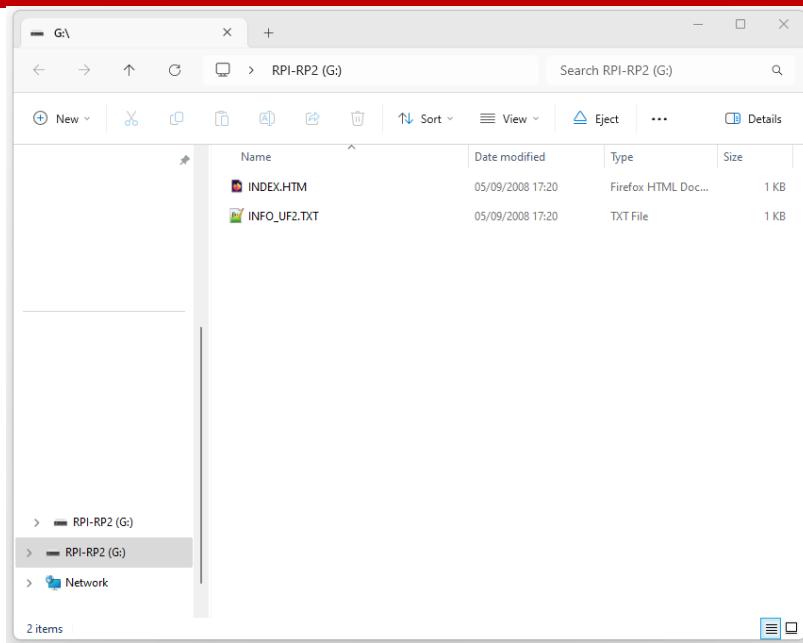
ARDUINO GIGA	(stable) 20240222-v1.22.2.uf2
ARDUINO NANO 33 BLE SENSE	(stable) 20231227-v1.22.0.uf2
ARDUINO NANO ESP32	(stable) 20231005-v1.21.0.uf2
ARDUINO NANO RP2040 CONNECT	(stable) 20230426-v1.20.0.uf2
ARDUINO NANO RP2040 CONNECT	(stable) 20220618-v1.19.1.uf2
ARDUINO NANO RP2040 CONNECT	(stable) 20220117-v1.18.uf2
ARDUINO NICLA VISION	(preview) 20240516-v1.23.0-preview.379.gcf5a8ea3.uf2
ARDUINO PORTENTA C33	(preview) 20240516-v1.23.0-preview.376.gc10a74b16.uf2
ARDUINO PORTENTA C33	(preview) 20240515-v1.23.0-preview.375.ga0d4fdcce.uf2
ARDUINO PORTENTA C33	(preview) 20240515-v1.23.0-preview.372.ge816b49c4.uf2

Una vez que se hay completado la descarga, puede continuar con la instalación del firmware.

La instalación del firmware se puede realizar principalmente de dos maneras: a través de una herramienta llamada **Arduino MicroPython Installer** o simplemente arrastrando y soltando el archivo del firmware en el almacenamiento **flash** de la tarjeta, que aparece como un dispositivo de almacenamiento externo masivo después de conectar la placa a una PC a través de un USB. Cableado.

Se usará este último método, por ser el más sencillo.

# MicroPython, Firts Contact



# MicroPython, First Contact

1.- Conectar la tarjeta Arduino Nano RP2040 Connect a una Computadora usando el conector micro USB.

Ahora pueden suceder dos cosas:

- a) Se reconoce un dispositivo de almacenamiento masivo y se abre el administrador de ventanas correspondiente.
- b) No pasa nada.

Poner la tarjeta Arduino Nano RP2040 Connect en modo de almacenamiento masivo (también llamado modo bootloader) es un paso fundamental para la instalación del firmware.

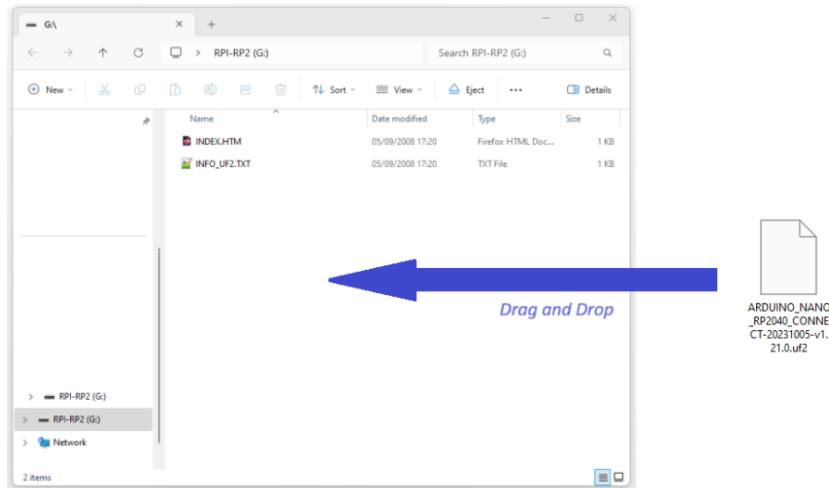
En caso de que no aparezca el almacenamiento masivo, se puede proceder de las dos formas siguientes:

- a) Presione el botón de reinicio dos veces secuencialmente con aproximadamente 1 segundo entre una presión y la siguiente, y si no funciona...
- b) Conecte físicamente el pin REC y GND en la placa y presione el botón de reinicio.

Una de estas dos formas debería configurar la placa en modo de gestor de arranque.

Una vez que se abra la ventana de almacenamiento **flash**, arrastre y suelte el firmware descargado previamente para iniciar la instalación

El dispositivo ahora está listo para ser programado en MicroPython.



## Obtener el IDE Arduino Lab for MicroPython

Una vez instalado el firmware, puedo descargar el IDE Arduino Lab, el cual proporciona una GUI muy fácil de usar.

Arduino Lab for MicroPython se puede descargar desde aquí:

<https://labs.arduino.cc/en/labs/micropython>

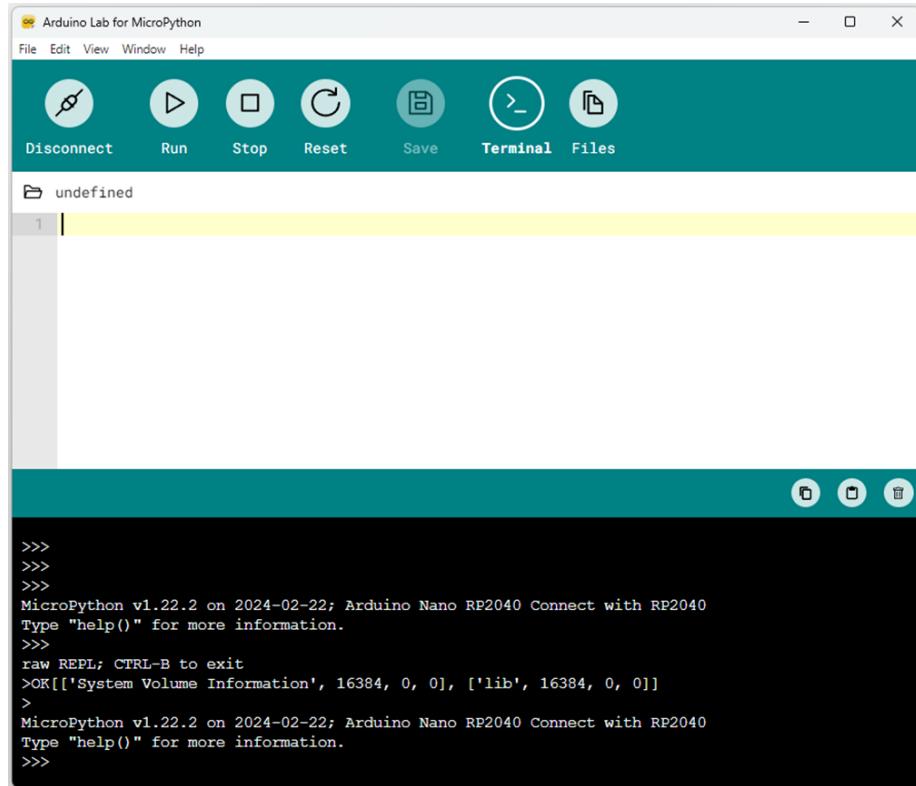


El archivo zip descargado contiene el IDE completo, que no requiere ningún proceso de instalación.

Simplemente se extrae todo el contenido en una carpeta y basta con abrir el archivo de la aplicación "Arduino Lab For MicroPython.exe". Una vez abierta, aparecerá la ventana IDE.

📁 locales	File folder
📁 resources	File folder
📘 Arduino Lab for Micropython.exe	Application
📄 chrome_100_percent.pak	PAK File
📄 chrome_200_percent.pak	PAK File
⚙️ d3dcompiler_47.dll	Application extension

# MicroPython, First Contact



La GUI de **Arduino Lab for MicroPython** es sencilla de usar y proporciona todas las funcionalidades necesarias para crear programas y programar la tarjeta.

En el siguiente capítulo, se mostrará cómo escribir un primer programa y ejecutarlo en la tarjeta Arduino Nano RP2040 Connect.

## Primer programa en MicroPython.

El primer programa MicroPython que se mostrará, hará parpadear un LED.

El siguiente código se puede copiar en **Arduino Lab**.

### A) Hola mundo tradicional

```
# Programa Hola mundo, imprime un mensaje en el REPL (consola)
print('\nHello, PyconLatam 2024!')
```

### B) Hola mundo para microcontroladores

```
import time
from machine import Pin

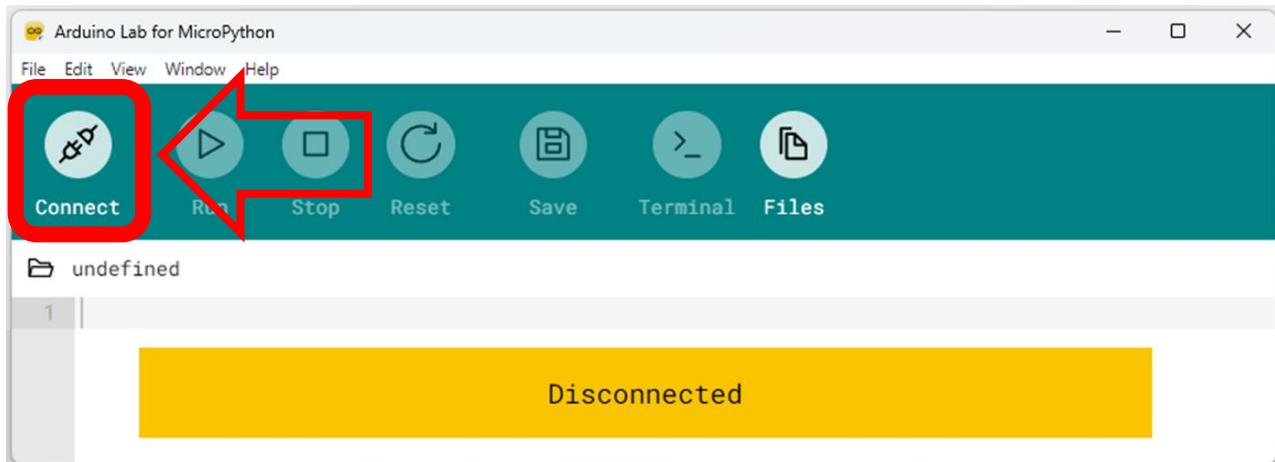
# Configuring the LED Pin
led = Pin(6, Pin.OUT)
while (True):
    # Switch on the LED
    led.on()
    time.sleep_ms(500)
    # Switch off the LED
    led.off()
    time.sleep_ms(500)
```

El código es bastante sencillo. En un primer momento se crea e inicializa un objeto “Pin” que hace referencia al **PIN (GPIO) 6 (LED\_BUILTIN)**, de la tarjeta **Arduino Nano RP2040 Connect**, el cual a su vez está conectado a un LED.

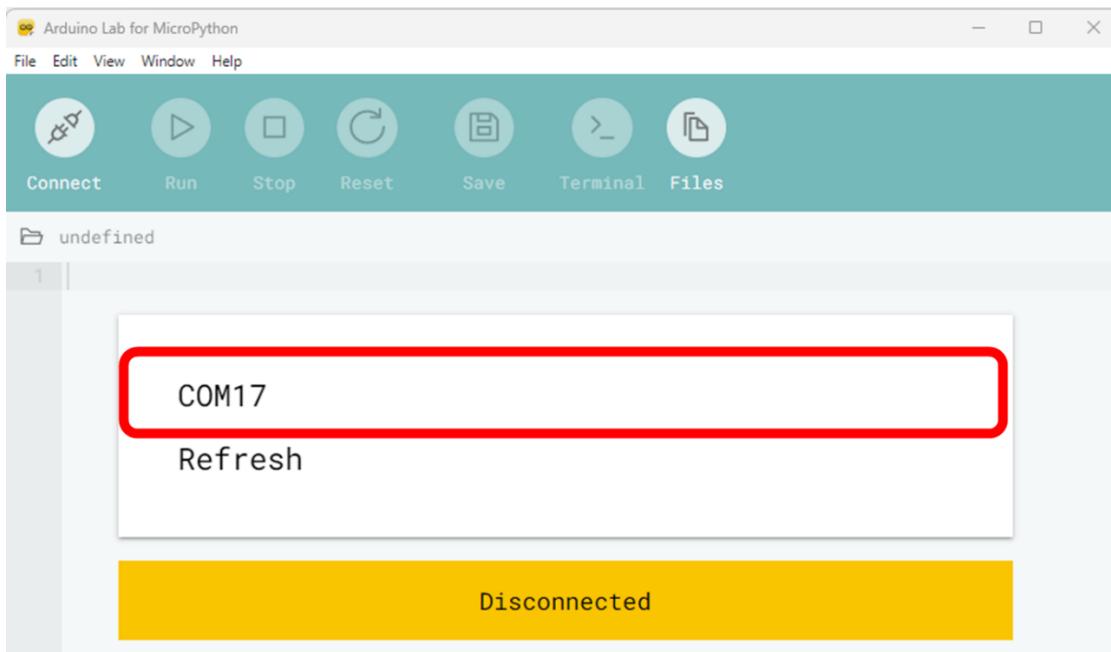
A continuación, se define un bucle infinito encendiendo y apagando el LED con un intervalo de tiempo de 500 ms.

Para probar el código, primero debe establecerse la conexión entre la tarjeta **Arduino Nano RP2040 Connect** y el IDE **Arduino Lab**, haciendo clic en el botón **Connect**, como se muestra en la Figura siguiente.

# MicroPython, First Contact

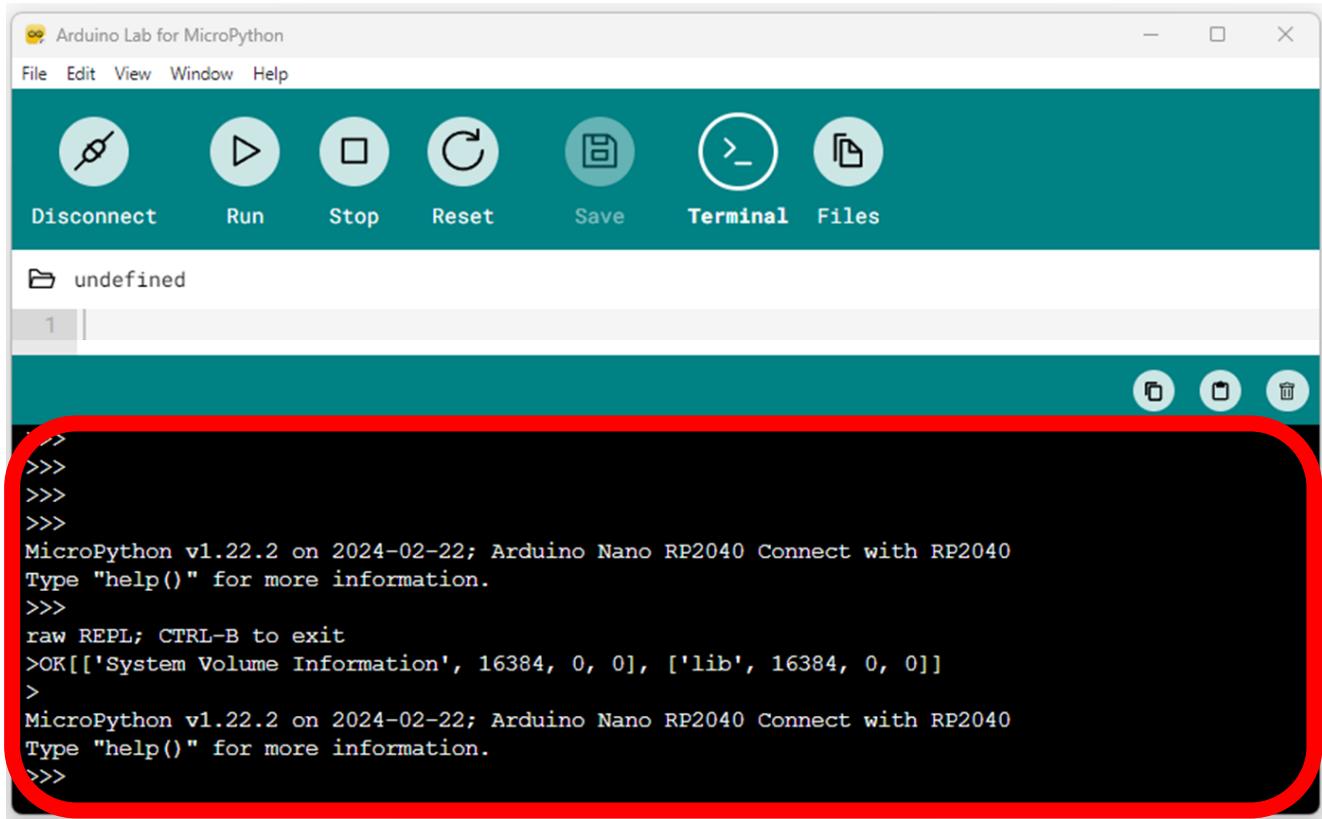


Seleccione el puerto COM correcto que aparece en la lista.



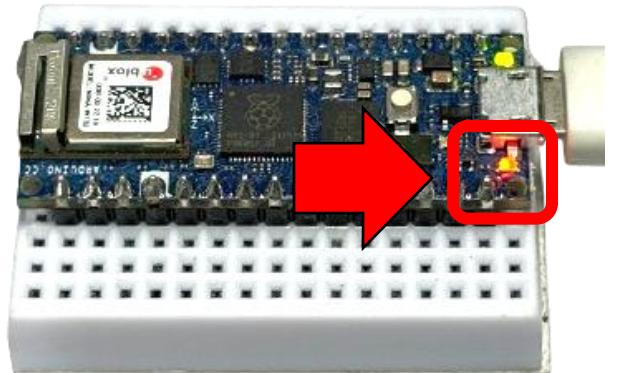
Se mostrará el siguiente resultado en la ventana de depuración de **Arduino Lab**.

# MicroPython, First Contact



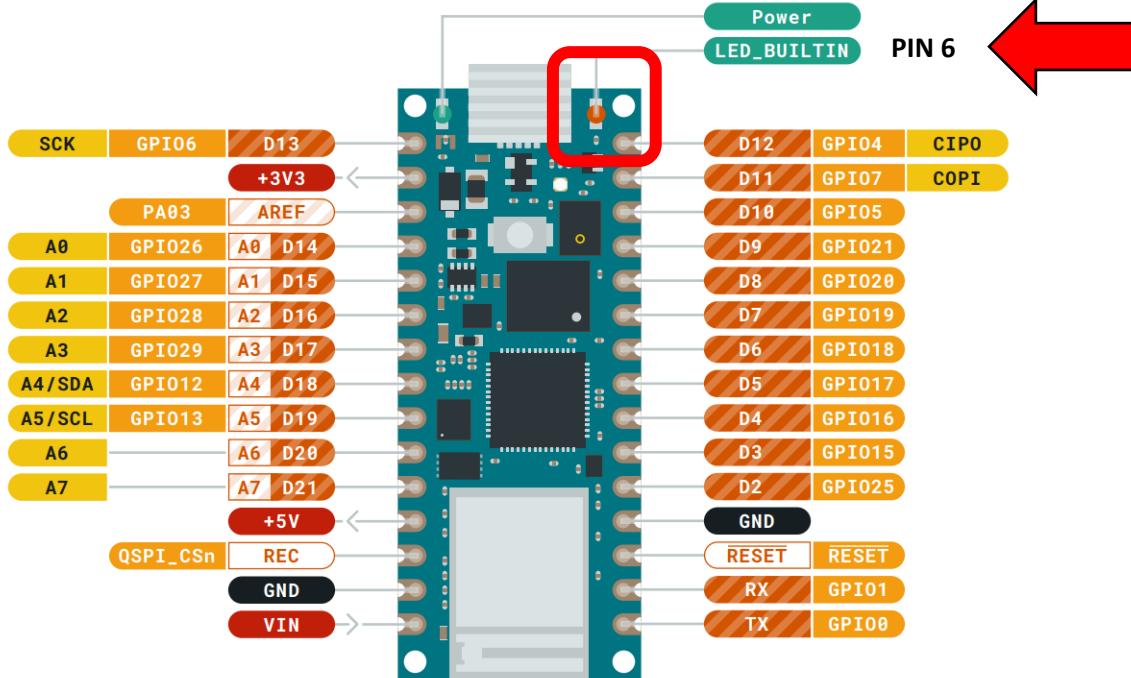
Ahora, haga clic en el botón "Run" para comenzar a ejecutar el código. El LED debería comenzar a parpadear.

The screenshot shows the Arduino Lab for MicroPython software window. The title bar says "Arduino Lab for MicroPython". The menu bar includes File, Edit, View, Window, and Help. Below the menu is a toolbar with icons for Disconnect, Run (highlighted with a red box and arrow), Stop, Reset, Save, Terminal, and Files. The main area has a folder icon labeled "undefined". Below that is a text input field containing "1". The central part is a terminal window displaying MicroPython code and its execution. The code imports time and machine modules, configures an LED pin, and enters a loop to toggle the LED every 500ms. The terminal also shows system information and the MicroPython version. At the bottom of the terminal window, there are three small icons: a square, a circle, and a triangle.

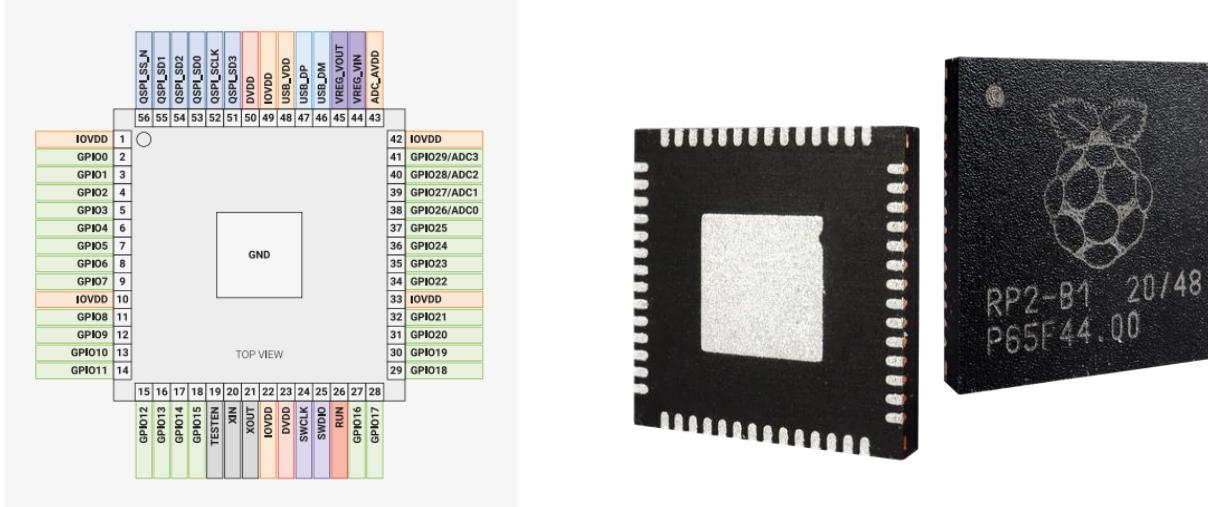


# MicroPython, First Contact

## Pinout Arduino Nano RP2040



## Pinout Microcontrolador RP2040



## Mostrar todos los pines disponibles

Nota.- Si desea ver los pines disponibles en su tarjeta, puede usar el siguiente comando:

```
from machine import Pin  
help(Pin.board)
```

```
>>> from machine import Pin  
>>> help(Pin.board)  
object <class 'board'> is of type type  
TX -- Pin(GPIO0, mode=ALT, pull=PULL_DOWN, alt=31)  
RX -- Pin(GPIO1, mode=ALT, pull=PULL_DOWN, alt=31)  
D12 -- Pin(GPIO4, mode=ALT, pull=PULL_DOWN, alt=31)  
D10 -- Pin(GPIO5, mode=ALT, pull=PULL_DOWN, alt=31)  
LED -- Pin(GPIO6, mode=OUT)  
D13 -- Pin(GPIO6, mode=OUT)  
D11 -- Pin(GPIO7, mode=ALT, pull=PULL_DOWN, alt=31)  
SDA -- Pin(GPIO12, mode=ALT, pull=PULL_DOWN, alt=31)  
SCL -- Pin(GPIO13, mode=ALT, pull=PULL_DOWN, alt=31)  
D3 -- Pin(GPIO15, mode=ALT, pull=PULL_DOWN, alt=31)  
D4 -- Pin(GPIO16, mode=ALT, pull=PULL_DOWN, alt=31)  
D5 -- Pin(GPIO17, mode=ALT, pull=PULL_DOWN, alt=31)  
D6 -- Pin(GPIO18, mode=ALT, pull=PULL_DOWN, alt=31)  
D7 -- Pin(GPIO19, mode=ALT, pull=PULL_DOWN, alt=31)  
D8 -- Pin(GPIO20, mode=ALT, pull=PULL_DOWN, alt=31)  
D9 -- Pin(GPIO21, mode=ALT, pull=PULL_DOWN, alt=31)  
D2 -- Pin(GPIO25, mode=ALT, pull=PULL_DOWN, alt=31)  
A0 -- Pin(GPIO26, mode=ALT, pull=PULL_DOWN, alt=31)  
A1 -- Pin(GPIO27, mode=ALT, pull=PULL_DOWN, alt=31)  
A2 -- Pin(GPIO28, mode=ALT, pull=PULL_DOWN, alt=31)  
A3 -- Pin(GPIO29, mode=ALT, pull=PULL_DOWN, alt=31)  
LED_RED -- Pin(EXT_GPIO0, mode=IN)  
LED_GREEN -- Pin(EXT_GPIO1, mode=IN)  
LED_BLUE -- Pin(EXT_GPIO2, mode=IN)  
D18 -- Pin(EXT_GPIO3, mode=IN)  
A4 -- Pin(EXT_GPIO3, mode=IN)  
D19 -- Pin(EXT_GPIO4, mode=IN)  
A5 -- Pin(EXT_GPIO4, mode=IN)  
A6 -- Pin(EXT_GPIO5, mode=IN)  
D20 -- Pin(EXT_GPIO5, mode=IN)  
A7 -- Pin(EXT_GPIO6, mode=IN)  
D21 -- Pin(EXT_GPIO6, mode=IN)  
>>> []
```

## Accediendo a sensores con MicroPython

Después de esta primera demostración del parpadeo de un LED, podemos profundizar en las funcionalidades extras que incorpora la tarjeta Arduino Nano RP2040 Connect y explorar cómo MicroPython puede interactuar de manera efectiva con los sensores incluidos.

Entre los periféricos disponibles, se encuentra el dispositivo LSM6DSOX, que es una pequeño IC (Circuito Integrado) que integra un acelerómetro y un giroscopio.

Los acelerómetros son sensores que miden la aceleración, indicando la tasa de cambio de velocidad de un objeto en una dirección específica. Al mismo tiempo, los giroscopios son dispositivos que miden o mantienen la orientación y la velocidad angular, proporcionando información sobre la rotación de un objeto.

A continuación, se muestra el ejemplo de código, que se puede copiar y pegar en **Arduino Lab** para leer los valores del sensor LSM6DSOX.

```
import time
from lsm6dsox import LSM6DSOX
from machine import Pin, I2C

# Initialize the LSM6DSOX sensor with I2C interface
lsm = LSM6DSOX(I2C(0, scl=Pin(13), sda=Pin(12)))

while True:
    # Read accelerometer values
    accel_values = lsm.accel()
    print('Accelerometer: x:{:>8.3f} y:{:>8.3f} z:{:>8.3f}'.format(*accel_values))

    # Read gyroscope values
    gyro_values = lsm.gyro()
    print('Gyroscope:   x:{:>8.3f} y:{:>8.3f} z:{:>8.3f}'.format(*gyro_values))

    print("")
    time.sleep_ms(100)
```

Ya se proporciona una biblioteca LSM6DSOX específica e integrada en el entorno **Arduino Lab**, lo que permite obtener fácilmente los datos del acelerómetro y giroscopio.

Los valores del sensor se leen a través del bus I2C, al que se accede a través de los pines 12 y 13, como se muestra en el fragmento de código.

## MicroPython, First Contact

La lectura del sensor se realiza en un ciclo de bucle infinito utilizando los métodos del objeto “accel()” y “gyro()”. Luego, los valores del sensor se imprimen en la consola de salida.

Proceda a copiar el código en el IDE de **Arduino Lab** y descárguelo en la tarjeta Arduino. El siguiente resultado se muestra en la consola de depuración, lo que demuestra la lectura exitosa de los valores del sensor.

```
Accelerometer values: x:0.07  y:0.00  z:1.01
Gyroscope values: x:0.24  y:0.61  z:-0.31

Accelerometer values: x:0.07  y:0.00  z:1.01
Gyroscope values: x:0.31  y:0.55  z:-0.24

Accelerometer values: x:0.07  y:0.00  z:1.01
Gyroscope values: x:0.12  y:0.61  z:-0.24
```

## IoT

Como se comentó al inicio de esta plática la tarjeta **Arduino Nano RP2040 Connect** ofrece grandes posibilidades para la creación de todo tipo de proyectos con IoT, a continuación, se mostrarán la capacidad de usar conectividad inalámbrica con WiFi.

### Conectividad WiFi

#### A) Búsqueda de redes wifi

```
# Scan Example
# This example shows how to scan for Wi-Fi networks.
import time, network

wlan = network.WLAN(network.STA_IF)
wlan.active(True)

print("\nPycon Latam 2024 Mazatlán")
print("\nScanning...")
while (True):
    scan_result = wlan.scan()
    for ap in scan_result:
        #print("Channel:%d RSSI:%d Auth:%d BSSID:%s SSID:%s"%(ap))
        print("SSID:%s BSSID:%s Channel:%d RSSI:%d Auth:%d hidden:%d "%(ap))
    print()
    time.sleep_ms(1000)
```

## B) Hacer una solicitud a una API

```
import urequests
import network
from time import sleep

# Request a random cat fact from the Meowfacts API
url = "https://meowfacts.herokuapp.com/"

WIFI_NETWORK='INFINITUM723C'
WIFI_PASSWORD='vK4xNVPxPX'

print("\n This program makes a random request to the Meow Facts API")
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect(WIFI_NETWORK, WIFI_PASSWORD)

# Pause required to configure Wi-Fi
print("\n Wait 5 seconds to WiFi Up")
sleep (5)

print("\n Connected to ",WIFI_NETWORK)
sleep (2)
# We should have a valid IP now via DHCP
print("\n WiFi Connected ", wlan.ifconfig())

# Pause required to get response
print("\n Wait 3 seconds for response")
sleep (3)

response = urequests.get(url)
print(response.text)
```

## Conclusiones

En conclusión, programar la tarjeta Arduino Nano Connect RP2040 con MicroPython abre un mundo de posibilidades para los desarrolladores que buscan un acercamiento al uso de microcontroladores, de una forma flexible y eficiente. La integración de MicroPython con el chip RP2040 simplifica el proceso de desarrollo, haciéndolo accesible incluso para quienes son nuevos en los sistemas embebidos.

Gracias a la existencia del ecosistema Arduino, la gran cantidad de bibliotecas creadas por la comunidad y ahora, se suma la familiar sintaxis de Python, por lo que en mi opinión la tarjeta Arduino Nano RP2040 Connect puede convertirse en una herramienta versátil para todo tipo de proyectos, desde aplicaciones de IoT, Salud, Agropecuaria, robótica etc.

Fuente:

<https://www.codemotion.com/magazine/backend/getting-started-with-micropython-on-arduino-nano-rp2040-connect/>

Autor: Matteo Trovo