



Bienvenido

Gracias por elegir los productos Freenove.

Cómo empezar

Cuando lea esto, deberá haber descargado el archivo ZIP de este producto.

Descomprímelo y obtendrás una carpeta que contiene tutoriales y archivos relacionados. Comience con este tutorial en PDF.

- ! Descomprima el archivo ZIP en lugar de abrir el archivo en el archivo ZIP directamente.
- ! No mueva, borre o cambie el nombre de los archivos de la carpeta que acaba de descomprimir.

Obtenga apoyo

¿Tiene problemas? No se preocupe. Consulte "TroubleShooting.pdf" o póngase en contacto con nosotros.

Si hay daños en el embalaje, problemas de calidad, preguntas sobre el uso, etc., sólo tiene que enviarnos un correo electrónico. Le responderemos en un día laborable y le daremos una solución.

support@freenove.com

Seguridad y precauciones

Siga las siguientes precauciones de seguridad cuando utilice o almacene este producto:

- Mantenga este producto fuera del alcance de los niños menores de 6 años.
- Este producto sólo debe utilizarse bajo la supervisión de un adulto, ya que los niños pequeños carecen del juicio necesario sobre la seguridad y las consecuencias del mal uso del producto.
- Este producto contiene piezas pequeñas y afiladas. Este producto contiene piezas conductoras de electricidad. Tenga cuidado con las piezas conductoras de la electricidad cerca o cerca de fuentes de alimentación, baterías y circuitos alimentados (con corriente).
- Cuando el producto se enciende, se activa o se prueba, algunas piezas se mueven o giran. Para evitar lesiones en las manos y los dedos, ¡manténgalos alejados de las piezas móviles!
- Es posible que un circuito mal conectado o en cortocircuito provoque un sobrecalentamiento. Si esto ocurre, desconecte inmediatamente la fuente de alimentación o retire las pilas y no toque nada hasta que se enfrie. Cuando todo esté seguro y frío, revise el tutorial del producto para identificar la causa.
- Sólo utilice el producto de acuerdo con las instrucciones y directrices de este tutorial, de lo contrario, las piezas podrían dañarse o usted podría resultar herido.
- Guarde el producto en un lugar fresco y seco y evite exponerlo a la luz solar directa.
- Después de su uso, apague siempre el aparato y retire o desenchufe las pilas antes de guardarlo.

¿Alguna preocupación? support@freenove.com

Sobre Freenove

Freenove ofrece productos y servicios electrónicos de código abierto en todo el mundo.

Freenove se compromete a ayudar a los clientes en su formación en robótica, programación y circuitos electrónicos para que puedan transformar sus ideas creativas en prototipos y productos nuevos e innovadores. Para ello, nuestros servicios incluyen, entre otros, los siguientes:

- Kits de proyectos educativos y divertidos para robots, coches inteligentes y drones
- Kits educativos para aprender sistemas de software robótico para Arduino, Raspberry Pi y micro:bit
- Surtidos de componentes electrónicos, módulos electrónicos y **herramientas** especializadas **Servicios de desarrollo y personalización de productos**

Puede encontrar más información sobre Freenove y obtener nuestras últimas noticias y actualizaciones a través de nuestro sitio web:

<http://www.freenove.com>

Copyright

Todos los archivos, materiales y guías de instrucción proporcionados son liberados bajo la [licencia Creative Commons Attribution NonCommercial-ShareAlike 3.0 Unported License](#). Se puede encontrar una copia de esta licencia en la carpeta que contiene los archivos del Tutorial y del software asociados a este producto.



Esto significa que puede utilizar estos recursos en sus propias obras derivadas, en parte o en su totalidad, pero **NO con la intención o el propósito de utilizarlos comercialmente**.

La marca y el logotipo de Freenove son derechos de autor de Freenove Creative Technology Co., Ltd. y no pueden utilizarse sin autorización escrita.



Otras marcas registradas y sus propietarios que aparecen en este documento:

Arduino® es una marca comercial de Arduino LLC (<https://www.arduino.cc/>).

Raspberry Pi® es una marca comercial de Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

Raspberry Pi Pico® es una marca comercial de Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

micro:bit® es una marca comercial de Micro:bit Educational Foundation (<https://www.microbit.org/>).

ESPRESSIF® y ESP32® son marcas comerciales de ESPRESSIF Systems (Shanghai) Co, Ltd (<https://www.espressif.com/>).

¿Alguna preocupación? support@freenove.com

Contenido

Bienvenido.....	1
Contenido	1
Prefacio	4
Raspberry Pi Pico	5
Raspberry Pi Pico W.....	9
Capítulo 0 Preparación (importante).....	13
0.1 Instalación de Thonny (importante).....	13
0.2 Configuración básica de Thonny.....	18
0.3 Grabación del firmware de Micropython (importante)	20
0.4 Thonny conectado a Raspberry Pi Pico	21
0.5 Códigos de prueba 	(importante) 25
0.6 Operación Thonny Common	36 06. Pegar el adhesivo en la Breadboard 41
Capítulo 1 LED (importante).....	42
Proyecto 1.1 	Blink 42
Proyecto 1.2 	Blink 51
Capítulo 2 Botones y LEDs.....	59
Proyecto 2.1 Botón y LED	60
Proyecto 2.2 MINI lámpara de mesa	64

Capítulo 3 Barra de LEDs

67

Proyecto	3.1	Luz	que	fluye
----------	-----	-----	-----	-------

67

Capítulo 4 Analógico y PWM

73

Proyecto	4.1	LED	que	respira
Luz	que	fluye	del 73 Proyecto 4.2 meteorito
Cómo	importar	un	módulo	personalizado
				de python

79

83

Capítulo 5 RGBLED

85

Proyecto	5.1	Luz	de	color	aleatorio
Proyecto	5.2		Color		degradado
Light.....					91

Capítulo 6 Buzzer

94

Proyecto					6.1
Timbre.....					94
Contenido					

Proyecto					Alertor
					100

Capítulo 7 Comunicación en serie 104

Proyecto	7.1	Impresión	en	serie
				104
Proyecto	7.2	Lectura	y	escritura
				en serie
				109

Capítulo 8 Convertidor AD

113

Proyecto	8.1	Leer	la	tensión	del	potenciómetro
----------	-----	------	----	---------	-----	---------------

113

Capítulo 9 Potenciómetro y LED	119
Proyecto	9.1
	Luz
	suave
119
Capítulo 10 Fotoresistencia y LED	123
Proyecto	10.1
	Control del LED a través de la fotorresistencia
123
Capítulo 11 Termistor	128
Proyecto	11.1
	Termómetro
128
Capítulo 12 Modos de trabajo WiFi (sólo para Pico W)	133
Proyecto	12.1
	Modo estación
133 Proyecto 12.2
Modo AP138
Proyecto	12.3
	Modo AP+Estación
142
Capítulo 13 TCP/IP (sólo para Pico W).....	146
Proyecto	13.1
	Como cliente
146
Proyecto	13.2
	Como servidor
159
Capítulo 14 LED de control con Web (sólo para Pico W).....	165
Proyecto	14.1
	Control_LED_por_Web
165
¿Qué es lo siguiente?	172

Prefacio

Raspberry Pi Pico es una placa diminuta, rápida y versátil construida con RP2040, un nuevo chip microcontrolador diseñado por Raspberry Pi en el Reino Unido. Empezar es tan fácil como arrastrar y soltar un archivo, es adecuado para principiantes y makers para desarrollar, diseñar e investigar.

Raspberry Pi Pico es programable en C/C++ y MicroPython. En este tutorial, utilizamos Micropython para desarrollar, que es un lenguaje muy fácil de aprender con código magro y simple, por lo tanto es muy adecuado para los principiantes a aprender y para el desarrollo secundario.

Si no has descargado el material relacionado con el tutorial de Raspberry Pi Pico, puedes descargarlo desde este enlace:

https://github.com/Freenove/Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico

En este tutorial, dividimos cada proyecto en 4 secciones:

- 1, Lista de componentes: ayuda a los usuarios a conocer y encontrar los componentes necesarios en cada proyecto.
- 2, Conocimiento de los componentes: permite aprender las características y el uso de los componentes.
- 3, Circuito: ayuda a construir el circuito para cada proyecto.
- 4, Código y anotación: facilita a los usuarios el aprendizaje del uso de Raspberry Pi Pico y el desarrollo secundario.

Después de completar los proyectos de este tutorial, también puedes combinar los componentes en diferentes proyectos para hacer tus propias casas inteligentes, coche inteligente, robot, etc., dando vida a tu imaginación y creatividad con Raspberry Pi Pico.

Si tiene algún problema o dificultad para utilizar este producto, póngase en contacto con nosotros para recibir asistencia técnica rápida y gratuita: support@freenove.com

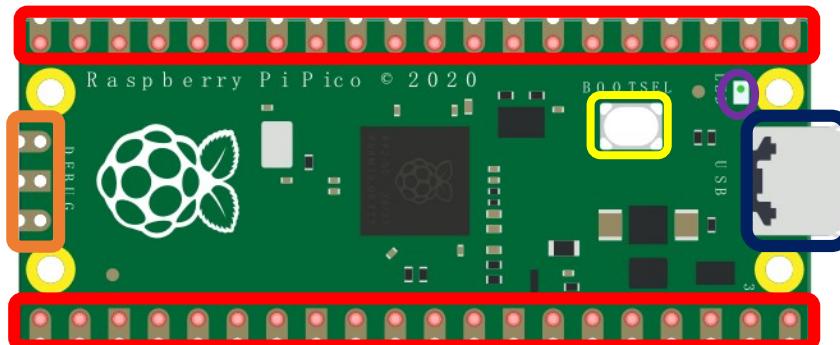
¿Alguna preocupación? support@freenove.com

Raspberry Pi Pico

Raspberry Pi Pico es un producto electrónico ligero, de tamaño diminuto y bajo precio. En la imagen de abajo podemos ver que sus recursos a bordo se han conectado a la interfaz de borde, que es muy adecuado para los entusiastas de la electrónica para utilizar en el bricolaje.



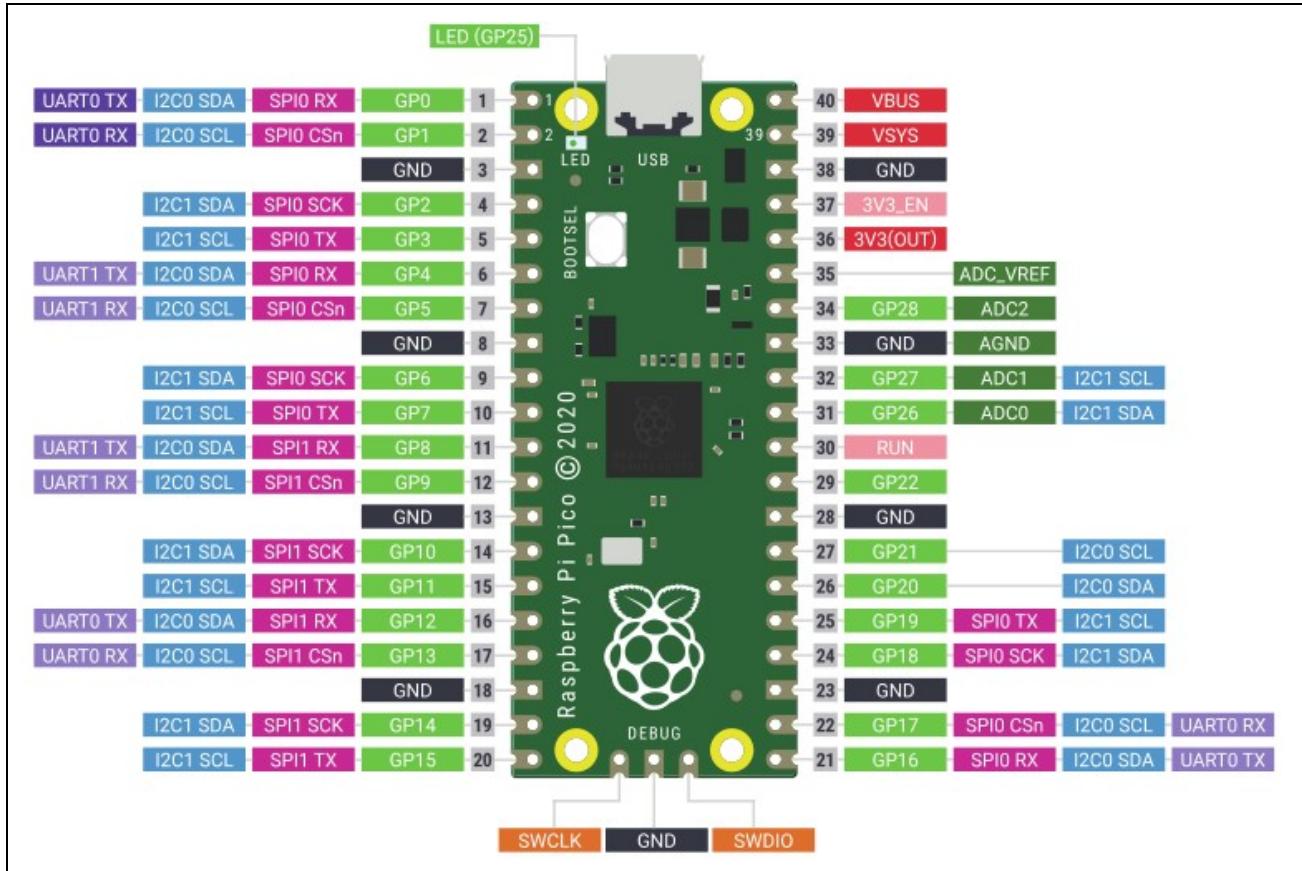
Las interfaces de hardware se distribuyen de la siguiente manera



Color del marco	Descripción
	Alfileres
	Botón BOOTSEL
	Puerto USB
	LED

	Depuración
--	------------

Definición de la función de los pines:



Color	Alfileres	Color	Alfileres
	GND		Potencia
	GPIO		ADC
	UART(default)		UART
	SPI		I2C
	Control del sistema		Depuración

Para más detalles : <https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf>

GND	Clavija de tierra
Potencia	VBUS(Voltaje microUSB)、VSYS(Entrada 2-5VDC)、3V3(Salida 3.3V)、3V3_EN(Habilita Pico)
Control del sistema	Ejecutar(Iniciar o desactivar el microcontrolador RP2040 o reiniciar)

¿Alguna preocupación? support@freenove.com

ADC	Raspberry Pi Pico tiene un total de 5 ADC con una resolución de 12 bits, que son ADC0(GP26), ADC1(GP27), ADC2(GP28), ADC3(GP29), ADC4 respectivamente. Entre ellos, ADC3(GP29) se utiliza para medir el VSYS en la placa Pico; ADC4 se conecta directamente al sensor de temperatura incorporado en el RP2040. ADC_VREF puede conectarse a un voltímetro externo de precisión como referencia del ADC. El pin ADC_GND se utiliza como punto de referencia para la conexión a tierra.
PWM	Hay 16 canales PWM en Raspberry Pi Pico, cada uno de los cuales puede controlar la frecuencia y el ciclo de trabajo de forma independiente. Los pines GPIO se comutan a la función PWM.
UART	Hay 2 UART: UART0, UART1.
SPI	Hay 2 SPI: SPI0, SPI1.
I2C	2 I2C: I2C0, I2C1.
Depuración	Se utiliza cuando se depura el código.

UART, I2C, SPI Pin defectuoso

UART

Función	Por defecto
UART_BAUDRATE	115200
UART_BITS	8
UART_STOP	1
UART0_TX	Pin 0
UART0_RX	Clavija 1
UART1_TX	Clavija 4
UART1_RX	Clavija 5

I2C

Función	Por defecto
Frecuencia I2C	400000
I2C0 SCL	Pin 9
I2C0 SDA	Clavija 8
I2C1 SCL	Clavija 7

I2C1 SDA

Clavija 6

SPI

Función	Por defecto
SPI_BAUDRATE	1000000
SPI_POLARITY	0
FASE_SPI	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI0_SCK	Clavija 6
SPI0_MOSI	Clavija 7
SPI0_MISO	Clavija 4
SPI1_SCK	Pin 10
SPI1_MOSI	Clavija 11
SPI1_MISO	Clavija 8

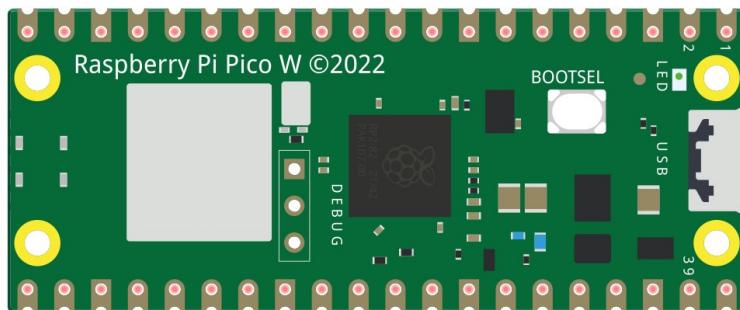
Para obtener información más detallada, consulte:

<https://datasheets.raspberrypi.org/pico/raspberry-pi-pico-python-sdk.pdf>

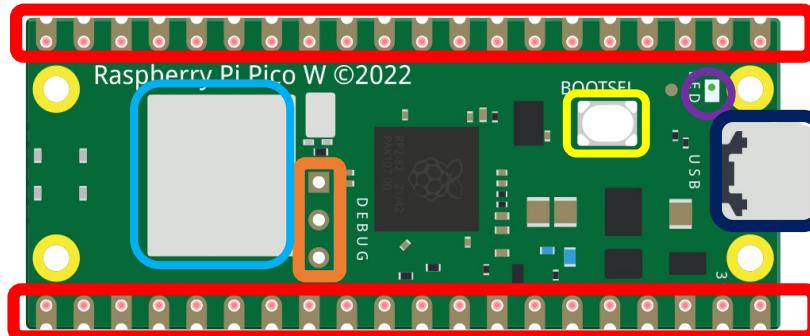
Raspberry Pi Pico W

Raspberry Pi Pico W se aplica a todos los capítulos de este tutorial.

Raspberry Pi Pico W añade CYW43439 como la función WiFi en la base de Raspberry Pi Pico. Se conecta al chip RP2040 a través de la interfaz SPI.

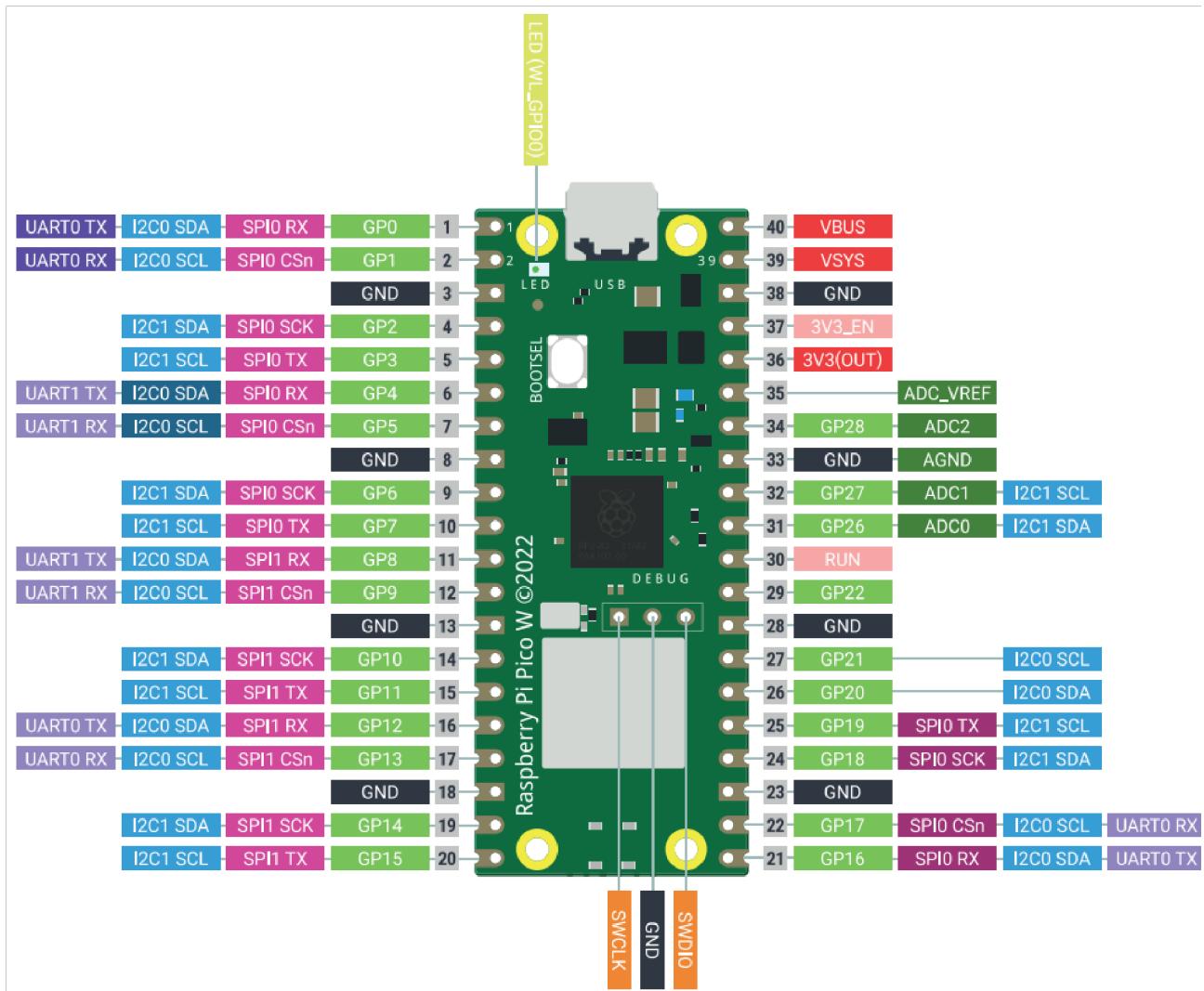


Las interfaces de hardware se distribuyen de la siguiente manera:



Color del marco	Descripción
	Alfileres
	Botón BOOTSEL
	Puerto USB
	LED
	Depuración
	Inalámbrico

Definición de la función de los pines:



Color	Alfileres	Color	Alfileres
Negro	GND	Rojo	Potencia
Verde	GPIO	Verde oscuro	ADC
Violeta	UART(default)	Lila	UART
Magenta	SPI	Cian	I2C
Rosa	Control del sistema	Naranja	Depuración

Para más detalles : <https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf>

UART, I2C, SPI, Clavija de Defecto Inalámbrica

En Arduino IDE, los pines por defecto del puerto serie son Pin0 y Pin1.

Nota: El puerto serial es virtualizado por el RP2040. Por lo tanto, cuando se utiliza el puerto serie, por favor, active la función de verificación de DTR. Puede funcionar bajo cualquier tasa de baudios.

UART

Función	Por defecto
UART_BAUDRATE	X
UART_BITS	8
UART_STOP	1
UART_TX	Pin 0
UART_RX	Clavija 1

I2C

Función	Por defecto
Frecuencia I2C	400000
I2C_SDA	Clavija 4
I2C_SCL	Clavija 5

SPI

Función	Por defecto
SPI_BAUDRATE	1000000
SPI_POLARITY	0
FASE_SPI	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Clavija 18
SPI_MOSI	Pin 19
SPI_MISO	Clavija 16
SPI_SS	Clavija 17

Inalámbrico

Función	Por defecto
WL_ON	GPIO23

WL_D	GPIO24
WL_CLK	GPIO29_ADC
WL_CS	GPIO25

Capítulo 0 Preparación (importante)

Antes de empezar a construir los proyectos, hay que hacer algunos preparativos, que son tan cruciales que no hay que saltárselos.

0.1 Instalación de Thonny (importante)

Thonny es una plataforma de software libre y de código abierto con un tamaño compacto, una interfaz sencilla, un funcionamiento simple y funciones ricas, lo que lo convierte en un IDE de Python para principiantes. En este tutorial, utilizamos este IDE para desarrollar Raspberry Pi Pico durante todo el proceso.

Thonny es compatible con varios sistemas operativos, como Windows, Mac OS y Linux.

Descargar Thonny

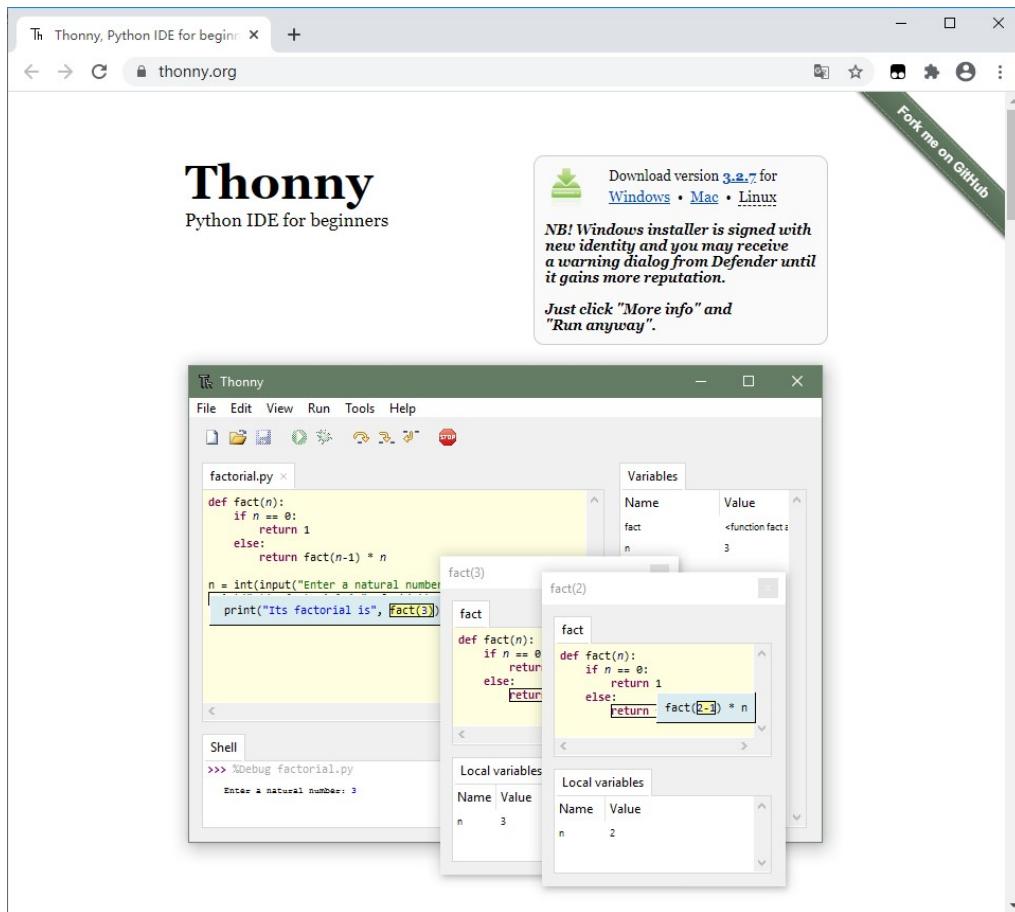
Página web oficial de Thonny: <https://thonny.org>

Repositorios de código abierto de Thonny: <https://github.com/thonny/thonny>

Siga las instrucciones del sitio web oficial para instalar Thonny o haga clic en los siguientes enlaces para descargarlo e instalarlo (seleccione el adecuado en función de su sistema operativo).

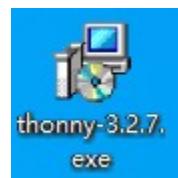
Funcionamiento	Enlaces/métodos de descarga
Sistema	
Windows	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.exe
Mac OS	https://github.com/thonny/thonny/releases/download/v3.2.7/thonny-3.2.7.pkg
Linux	<p>La última versión:</p> <p>Paquete binario para PC (Thonny+Python): bash <(wget -O - https://thonny.org/installer-for-linux)</p> <p>Con pip: pip3 install thonny</p> <p>Paquetes de distribución (pueden no ser la última versión):</p> <p>Debian, Rasbian, Ubuntu, Mint y otros: sudo apt install thonny</p> <p>Fedora: sudo dnf install thonny</p>

También puedes abrir "[Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python_Software](#)", lo hemos preparado de antemano.



Instalación en Windows

El ícono de Thonny después de la descarga es el siguiente. Haga doble clic en "thonny-3.2.7.exe".

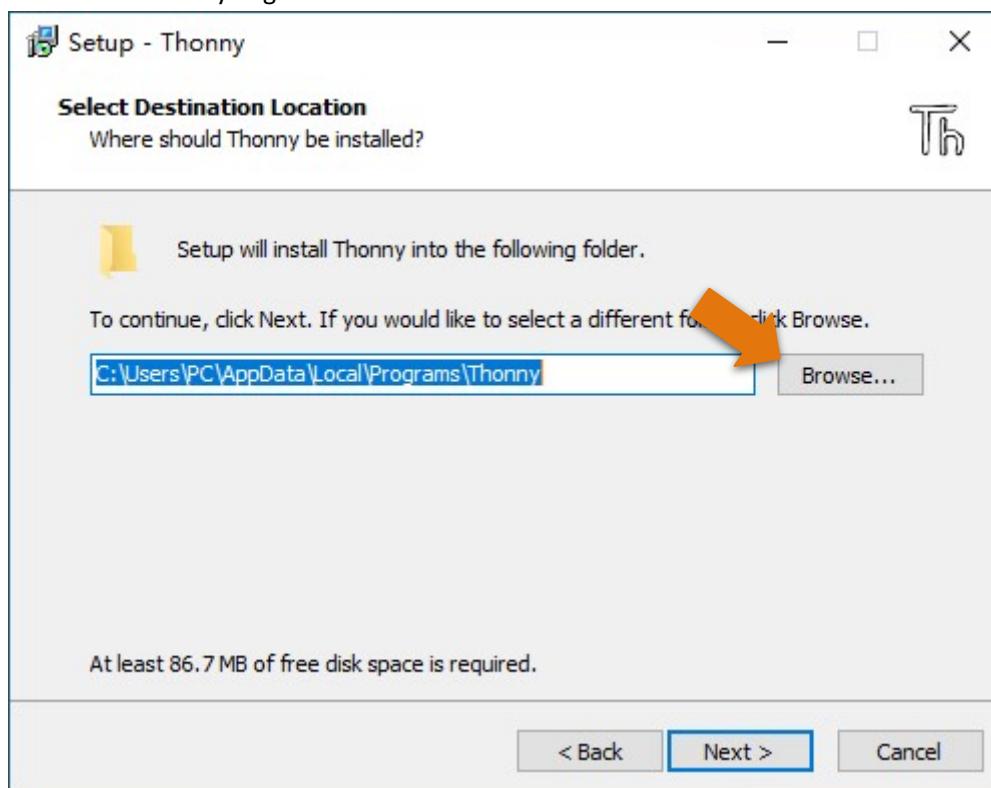


Si no está familiarizado con la instalación de programas informáticos, puede seguir haciendo clic en "Siguiente" hasta que se complete la instalación.

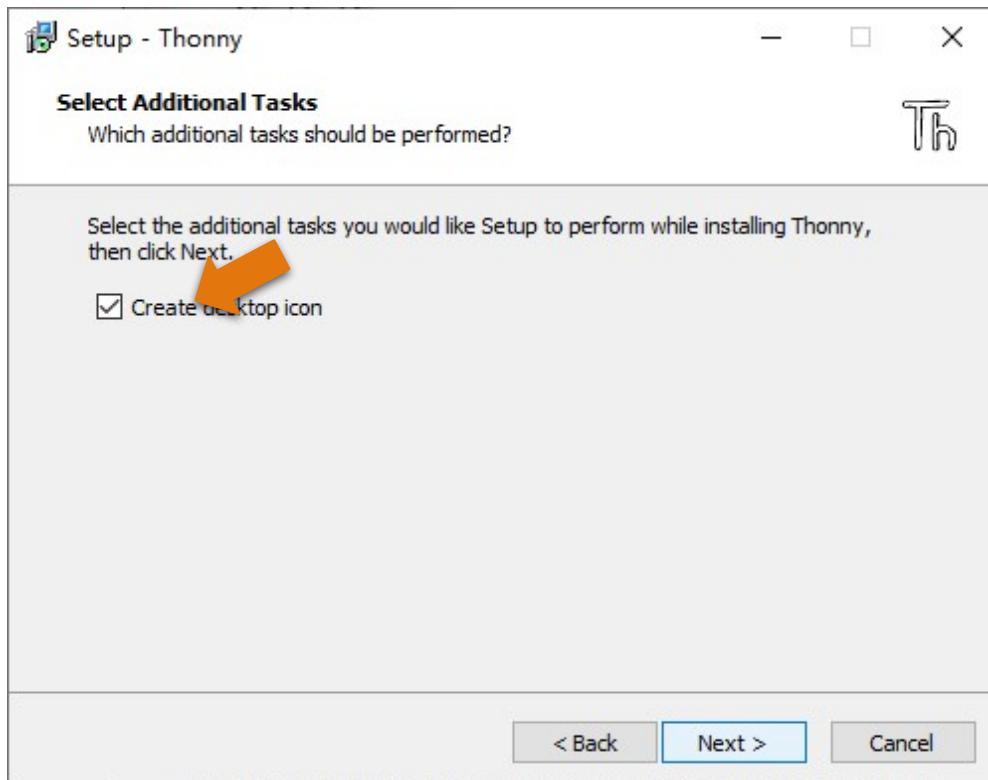


Si desea cambiar la ruta de instalación de Thonny, puede hacer clic en "Examinar" para modificarla. Después de seleccionar la ruta de instalación, haga clic en "Aceptar".

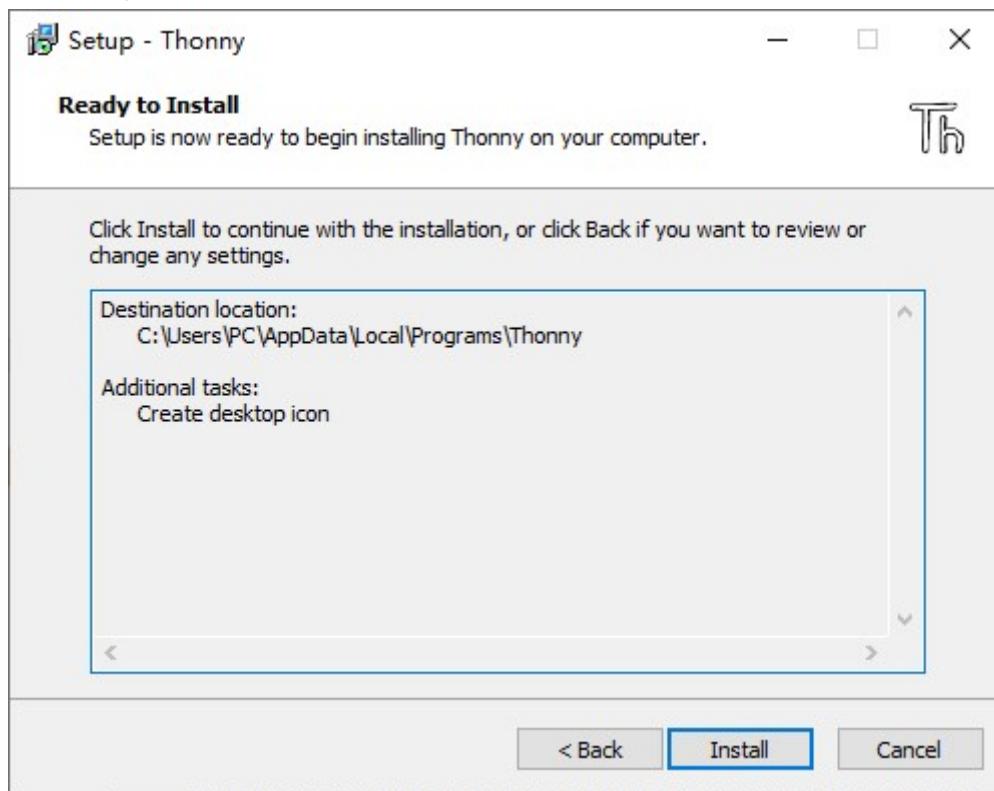
Si no quieres cambiar _____ y haga clic en



Marque la opción "Crear ícono en el escritorio" y entonces se generará un acceso directo en su escritorio para facilitarle la apertura de Thonny más adelante.

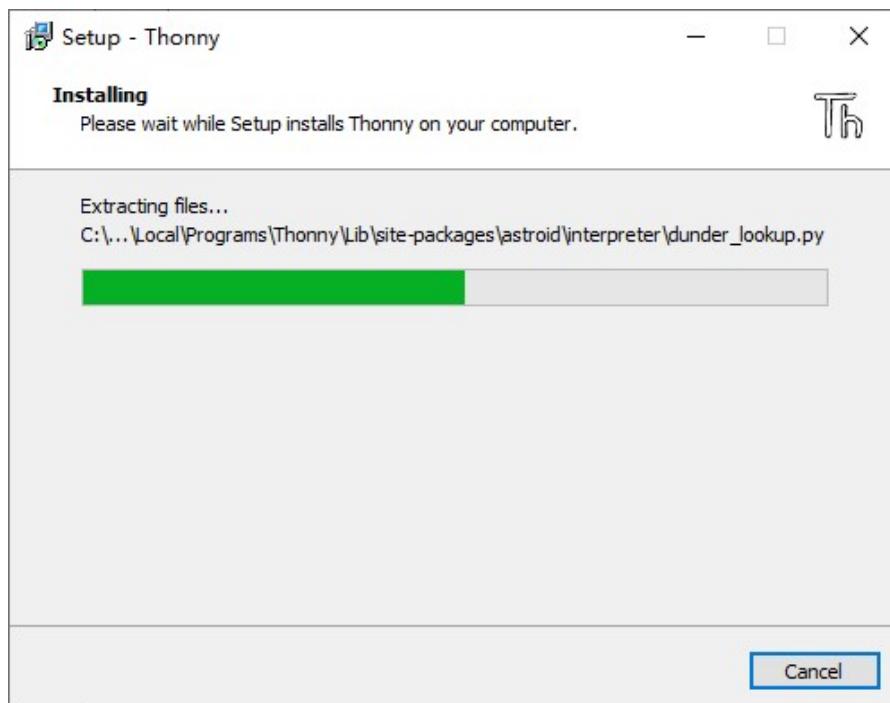


Haga clic en "instalar" para instalar el



Durante el proceso de instalación, sólo tiene que esperar a que la instalación se complete, y no debe hacer clic en "Cancelar", de lo contrario Thonny no se instalará.

¿Alguna preocupación? support@freenove.com



Una vez que vea la interfaz como la siguiente, Thonny se ha instalado con éxito.

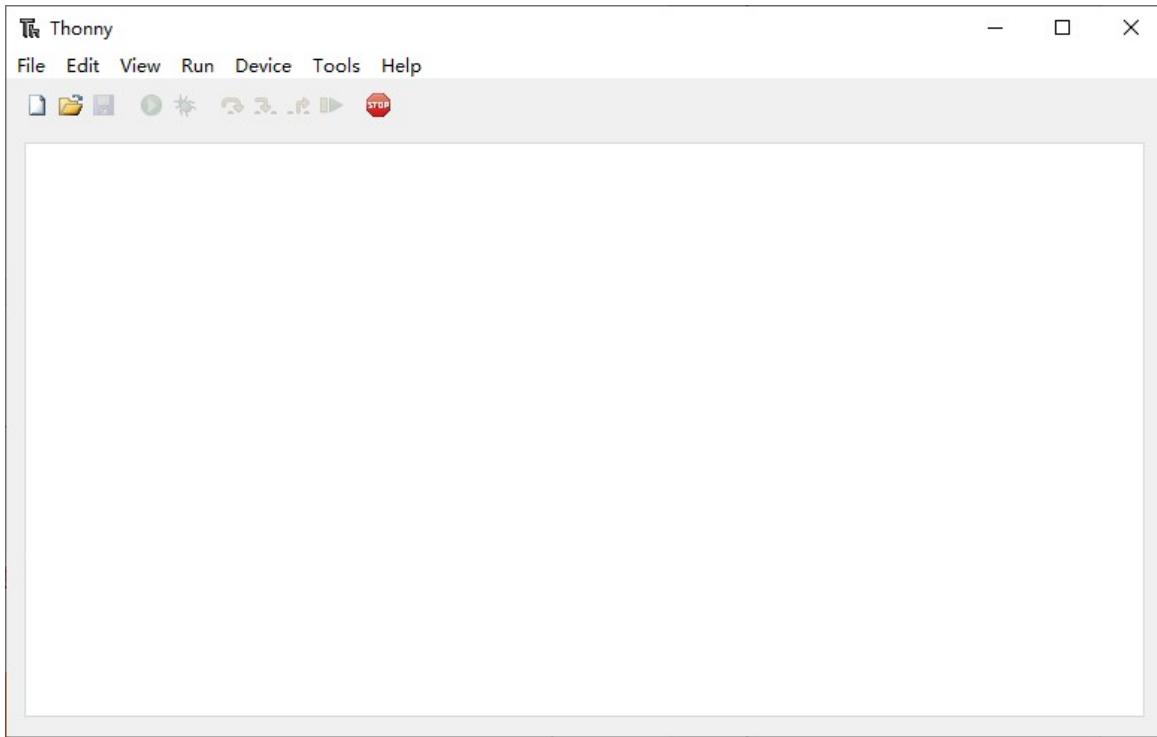


Si has marcado la opción "Crear ícono en el escritorio" durante el proceso de instalación, podrás ver el siguiente ícono en tu escritorio.

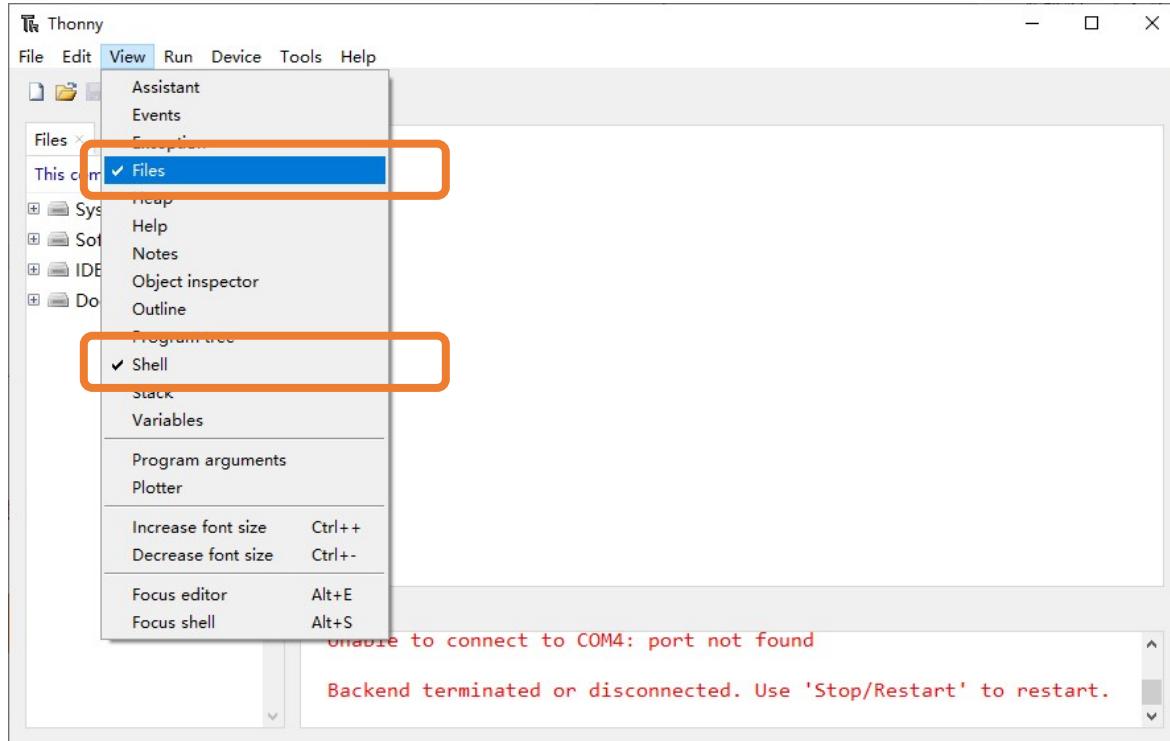


0.2 Configuración básica de Thonny

Click the desktop icon of Thonny and you can see the interface of it as follows:



Seleja "Ver" "Archí" y "Shell".





0.3 Grabación del firmware de Micropython (importante)

Para ejecutar programas de Python en Raspberry Pi Pico, necesitamos grabar un firmware en Raspberry Pi Pico primero.

Descarga del firmware de Micropython

Opción 1

Sitio web oficial de Raspberry Pi Pico: <https://www.raspberrypi.com/documentation/microcontrollers/>
1, Haga clic en Micropython.

The screenshot shows the official Raspberry Pi Documentation website. At the top, there are three navigation tabs: 'Computers', 'Accessories', and 'Microcontrollers'. The 'Microcontrollers' tab is currently selected, indicated by a red underline. Below the tabs, there are several items listed:

- RP2040**: Our first microcontroller device. It features a small chip icon and a brief description.
- Raspberry Pi Pico and Pico W**: Support for Raspberry Pi Pico, Pico H, and Pico W. It features a small chip icon and a brief description.
- MicroPython**: Getting started with MicroPython. This item is highlighted with a large red rounded rectangle.
- The C/C++ SDK**: Getting started with the C/C++ SDK. It features a hexagonal icon with a 'C' and a brief description.
- PIP**: The Product Information Portal (PIP) for Raspberry Pi compliance documents. It features a document icon and a brief description.
- Datasheets**: The Datasheets site for PDF-based documentation. It features a document icon and a brief description.

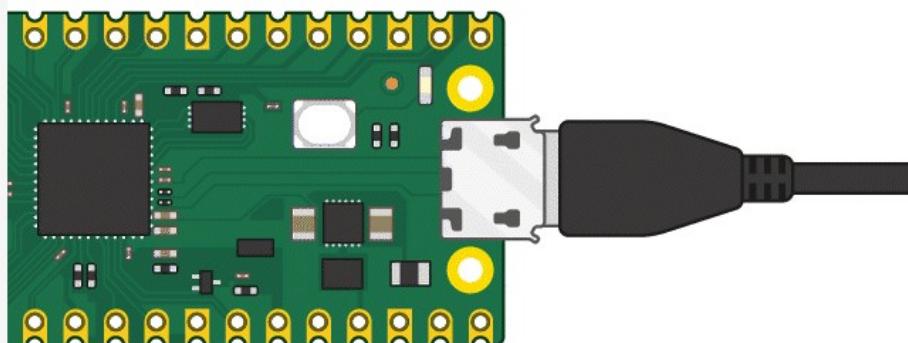
2, en la nueva interfaz, busque lo siguiente contenido y descargar el UF2 correspondiente según a tu versión Pico .

Drag-and-Drop MicroPython

[Edit this on GitHub](#)

You can program your Pico by connecting it to a computer via USB, then dragging and dropping a file onto it so we've put together a downloadable UF2 file to let you install MicroPython more easily.

2



Download the correct MicroPython UF2 file for your board:

- [Raspberry Pi Pico](#)
- [Raspberry Pi Pico W \(with urequests and upip preinstalled\)](#)

Then go ahead and:

1. Push and hold the BOOTSEL button and plug your Pico into the USB port of your Raspberry Pi or other computer. Release the BOOTSEL button after your Pico is connected.
2. It will mount as a Mass Storage Device called RPI-RP2.

Si está utilizando una placa Pico , por favor haga clic en Raspberry Pi Pico para descargar eso _

Si está utilizando una placa Pico W , por favor haga clic en Raspberry Pi Pico W para descargar eso _

Opción 2

Si tú no poder descargar eso adeudado a la red tema o otro razones , puedes usar el uno nosotros tener preparado , que se encuentra en la siguiente ruta de archivo :

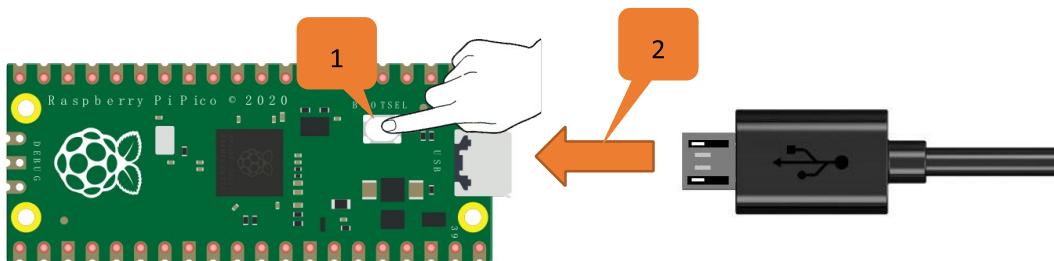
[**Freenove_Basic_Starter_Kit_para_Raspberry_Pi_Pico / Python_Firmware**](#)

Grabación de un firmware de Micropython

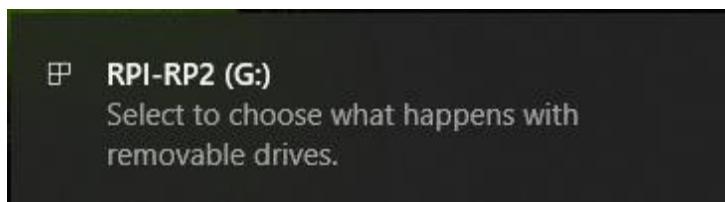
Nota: Pico y Pico W graban el firmware en el mismo camino _ de pico mapa es usó aquí como un introducción _

1. Conecte un cable USB a tu computadora

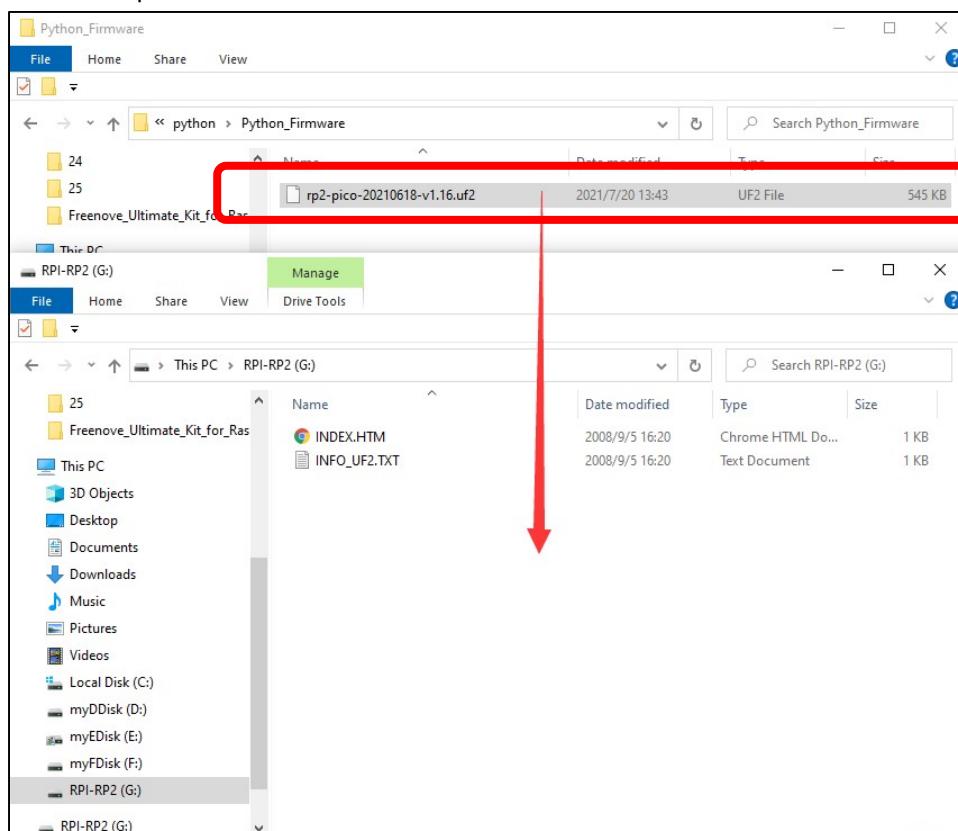
2. Mantenga presionado el botón BOOTSEL en Raspberry Pi Pico y conéctese eso a su computadora con el cable USB.



3. Cuando la conexión tiene éxito , lo siguiente información aparecerá en _ tu computadora



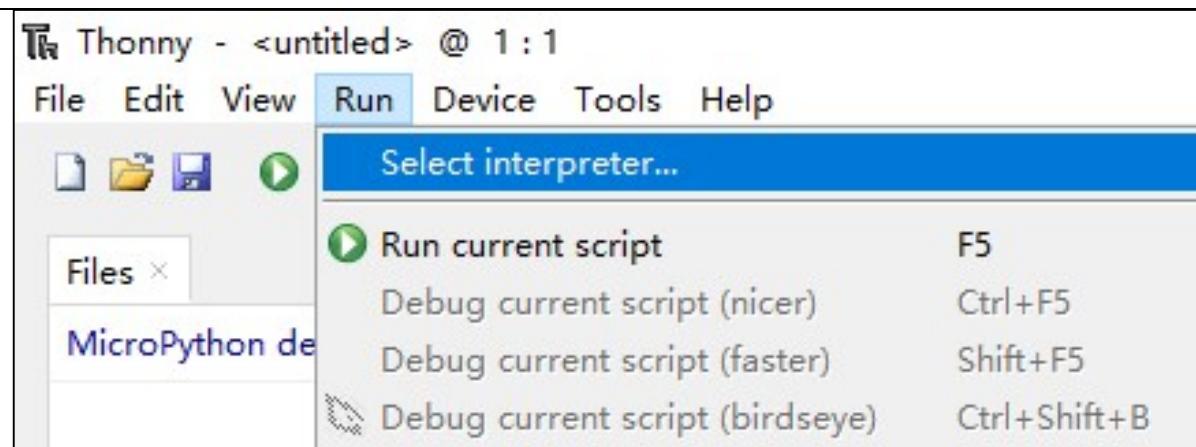
4. Copie el archivo (rp2-pico-20210618-v1.16.uf2) a RPI -RP2 y esperearlo a terminar , como copie el archivo a un disco U.



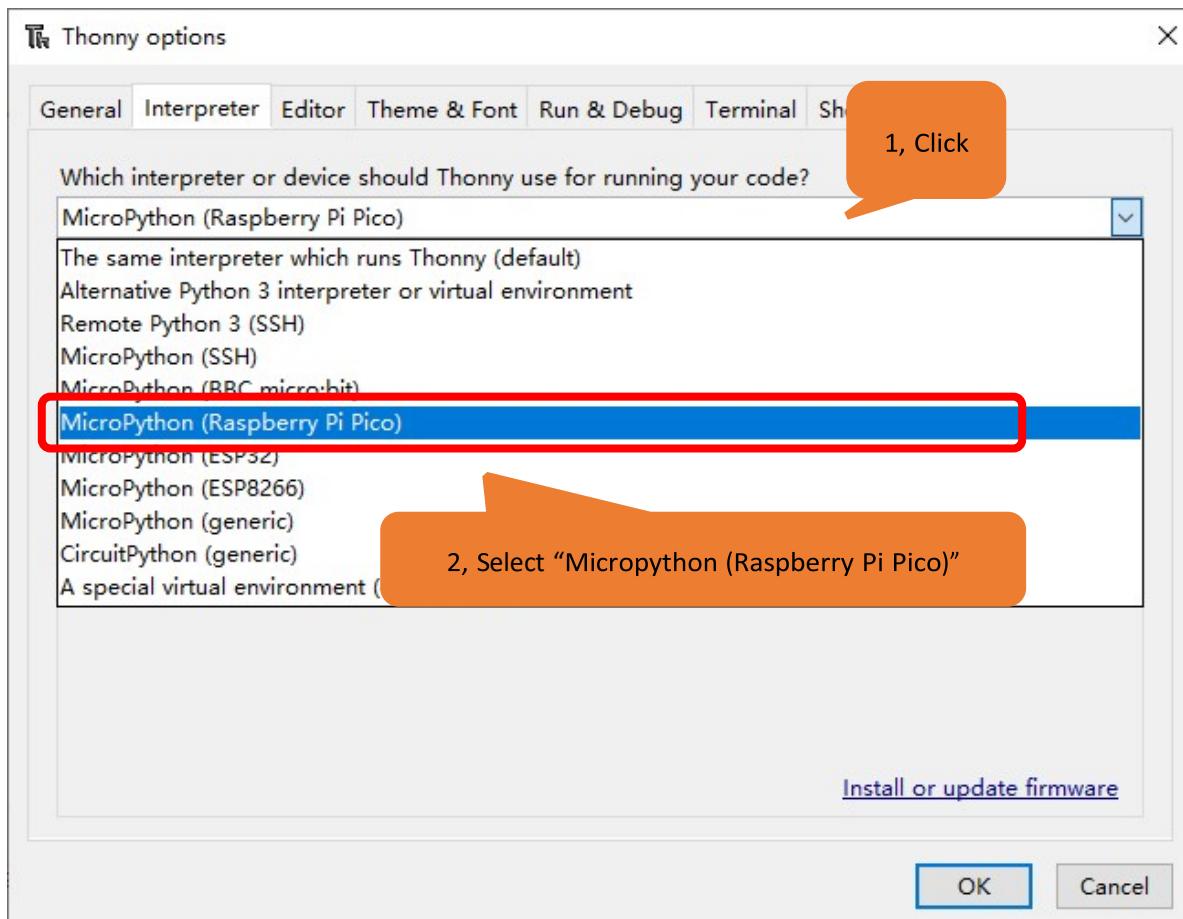
5. Cuando finaliza el firmware programación , Raspberry Pi Pico reiniciar automáticamente _
Después de eso , puede ejecutar Micropython .

0.4 delgado Conectado a Raspberry Pi Pico

1. Abra Thonny , haga clic en "ejecutar" y seleccione " Seleccionar intérprete ..."



2. Select “Micropython (Raspberry Pi Pico)”.

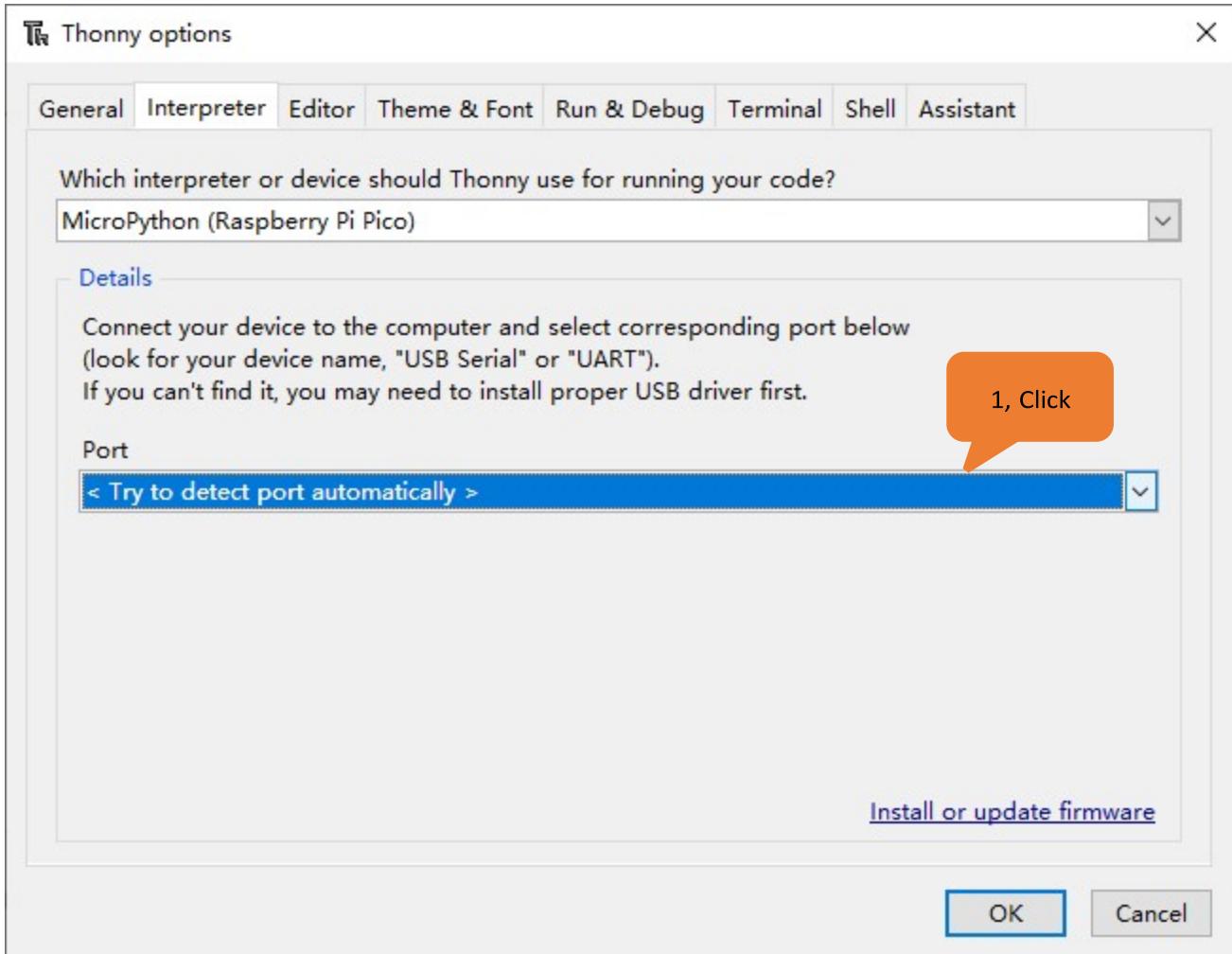


3. Seleccione "USB-SERIAL (COMx)", el número x de COMx mayo varía entre diferentes computadoras _ Tú solamente necesitar a hacer Por supuesto seleccionando USB-SERIAL (COMx).

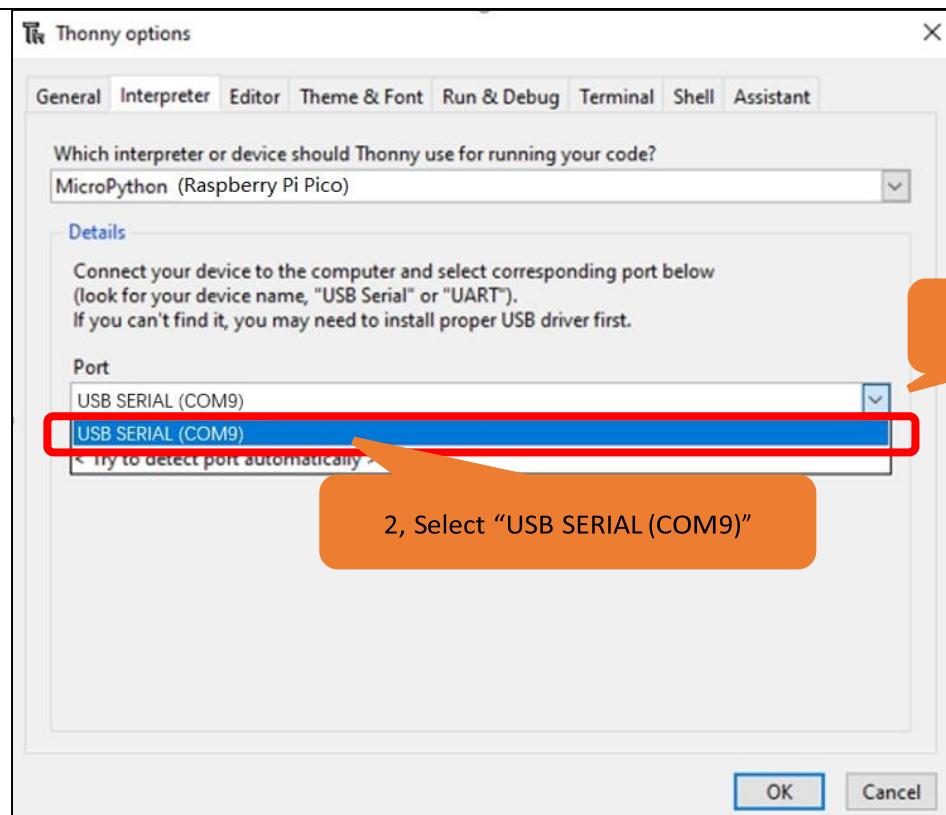
Cómo para determinar el puerto en cual tu Raspberry Pi Pico se comunica con tu computadora?

Paso 1: Cuando Pico **no lo hace** conectar a la computadora, abra Thonny , haga clic en "Ejecutar", seleccione " Seleccionar intérprete " y luego aparecerá un cuadro de diálogo , haga

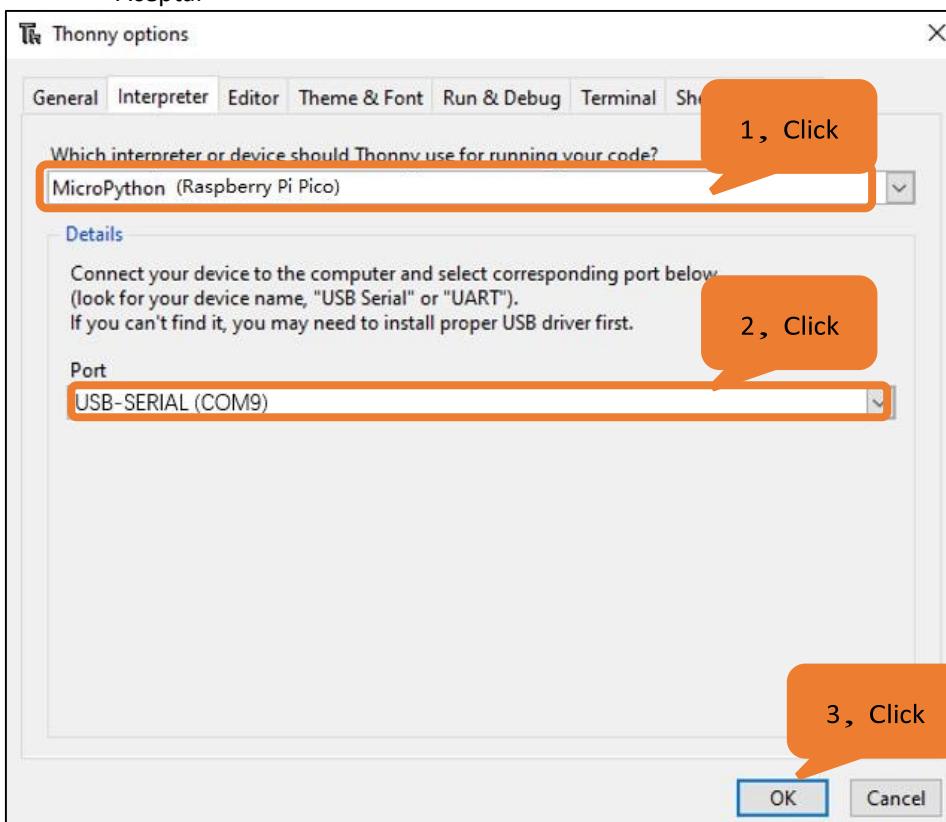
clic en “Puerto” y podrá verificar los puertos corrientemente conectado a su computadora, como se muestra a continuación :



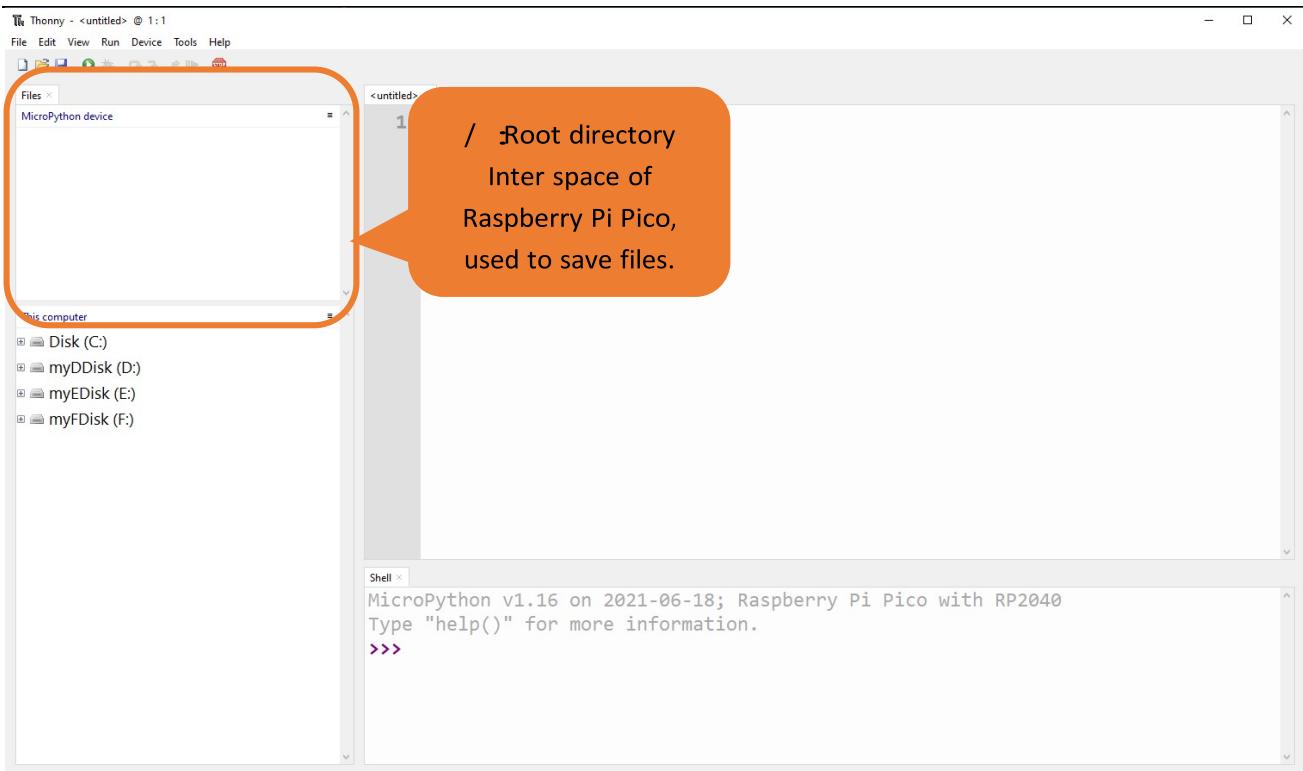
Paso 2: cierra el cuadro de diálogo . Conecta Pico a su computadora, haga clic en "Ejecutar " nuevamente y seleccione " Seleccionar intérprete ". Haga clic en "Puerto" en la ventana emergente y verifique el actual puertos _ Ahora allá es un nuevo adicional puerto , con que pico comunica con la computadora



4. Después de seleccionar " Micropython (Raspberry Pi Pico)" y el puerto , haga clic en "Aceptar"



5. cuando lo siguiente mensaje muestra en Thony , eso indica Thonny ha logrado conectado a Pico.

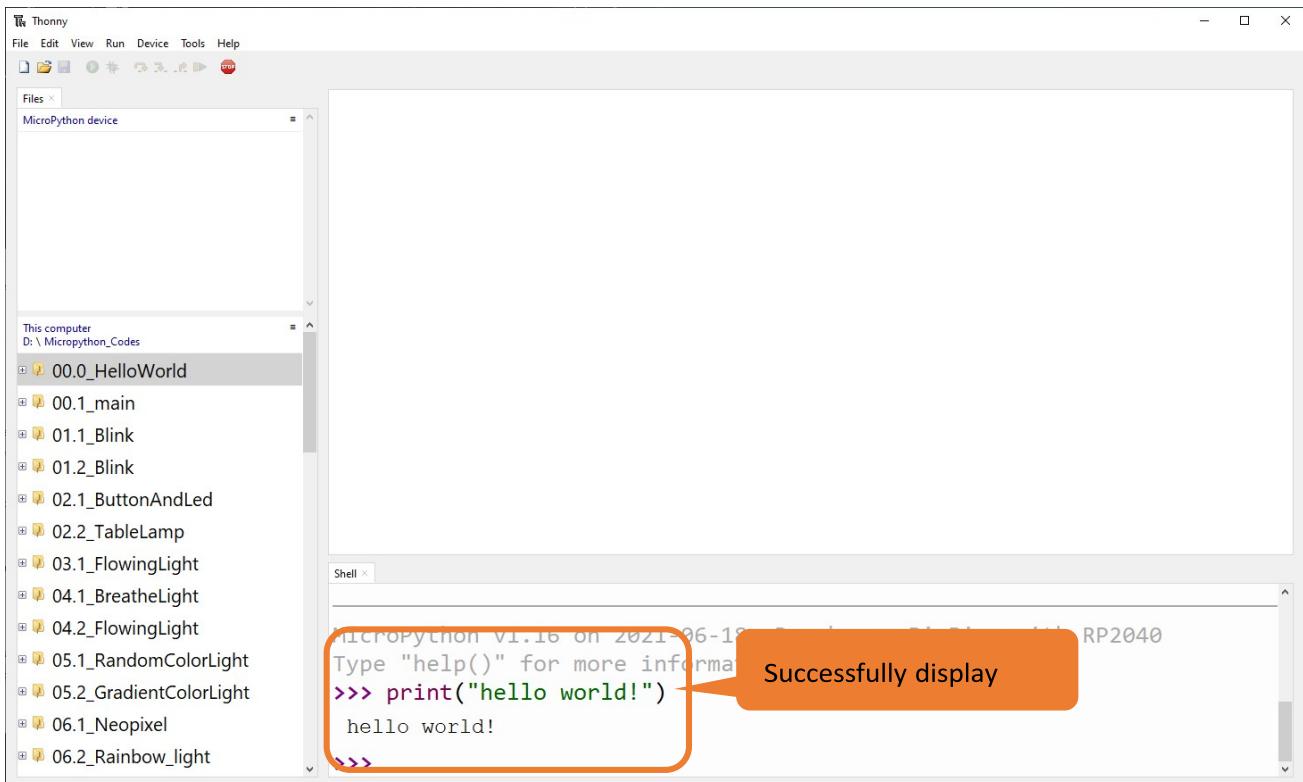


So far, all the preparations have been made.

0.5 Pruebas códigos (Importante)

Comando de prueba de Shell

Enter "print("hello world!")" in "Shell" and press Enter.



Capítulo 0 Conseguir Listo (Importante)

Correr en línea

Para ejecutar Raspberry Pi Pico en línea, usted necesitar conectar eso a la computadora Los usuarios pueden usar Thonny para compilar o depurar programas _

ventajas :

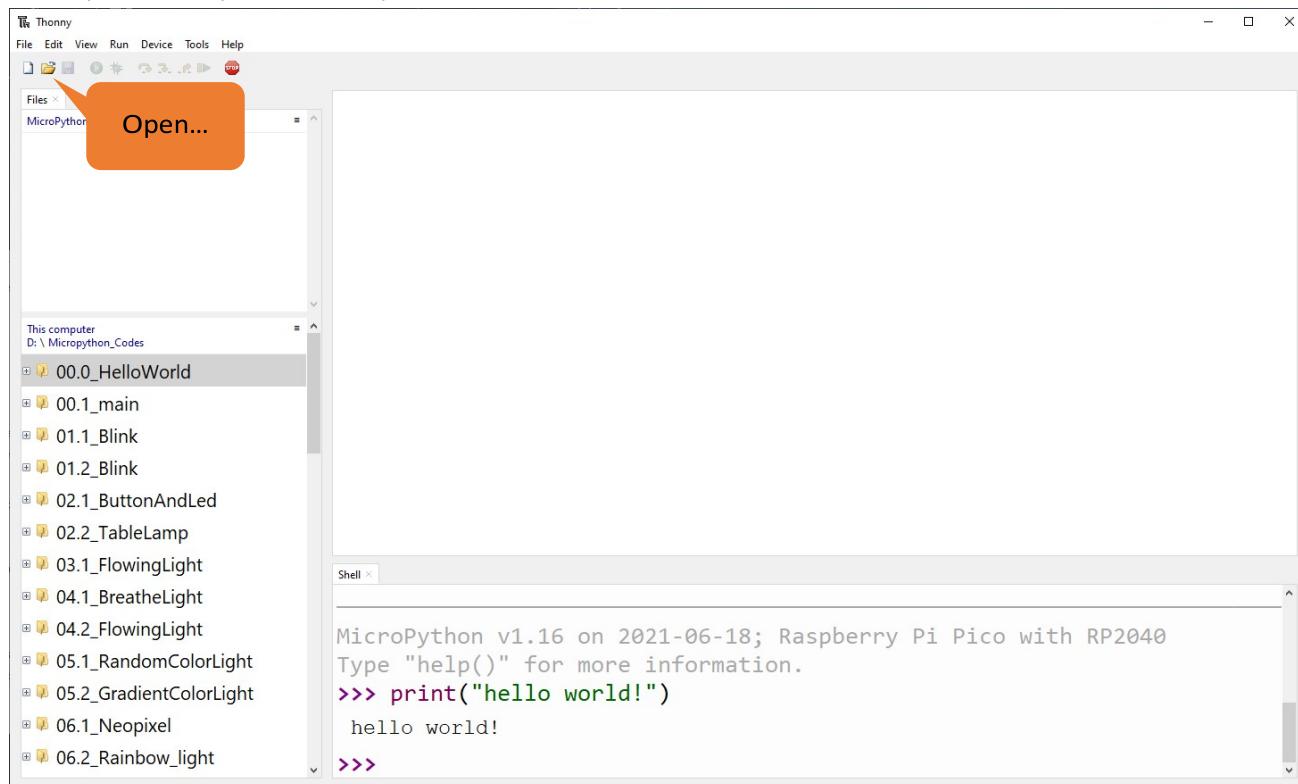
1. Los usuarios pueden usar Thonny para compilar o depurar programas _
2. A través de la ventana "Shell" , los usuarios pueden leer la información de error y generar resultados generado durante la ejecución del programa y consulta relacionado función información en línea para ayuda mejorar el programa .

Desventajas :

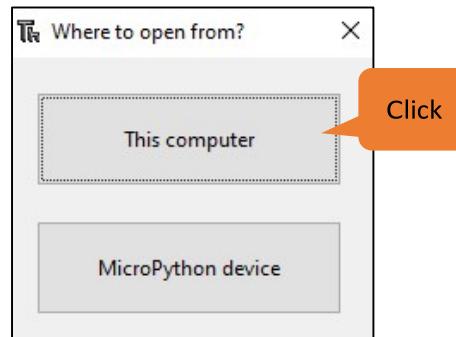
1. Para ejecutar Raspberry Pi Pico en línea, usted tener estar conectado _ a una computadora y ejecutar con thony _
2. Si Raspberry Pi Pico se desconecta desde la computadora, el programa no correrá de nuevo cuando ellos reconectar a cada otro _

Operación

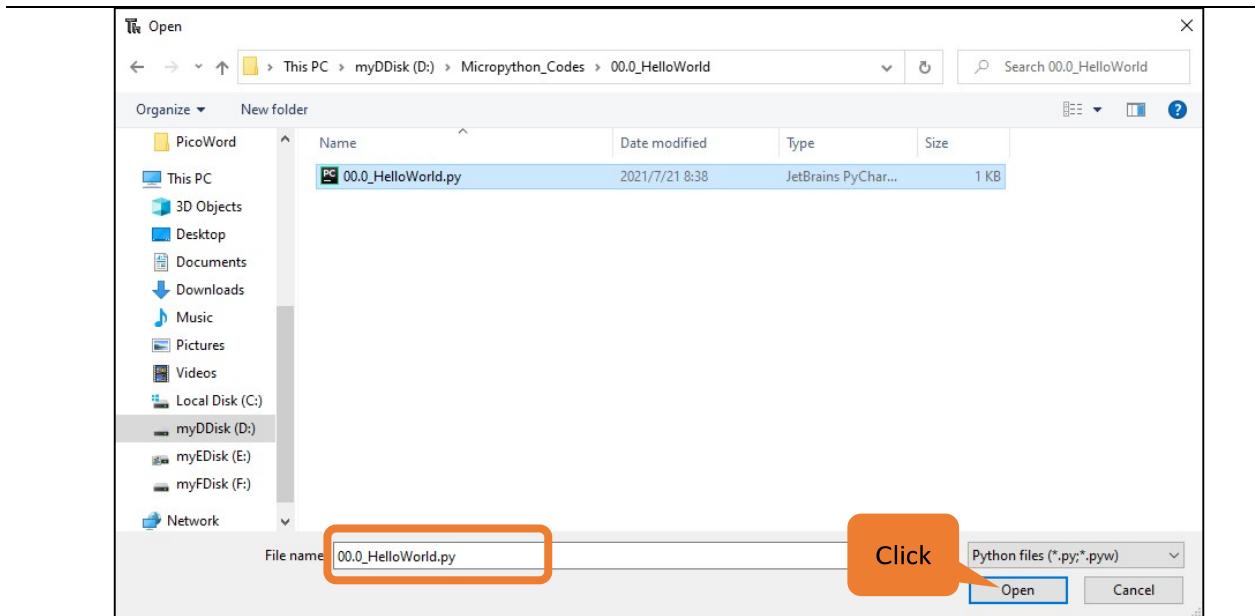
1. Open Thonny and click “Open...”.



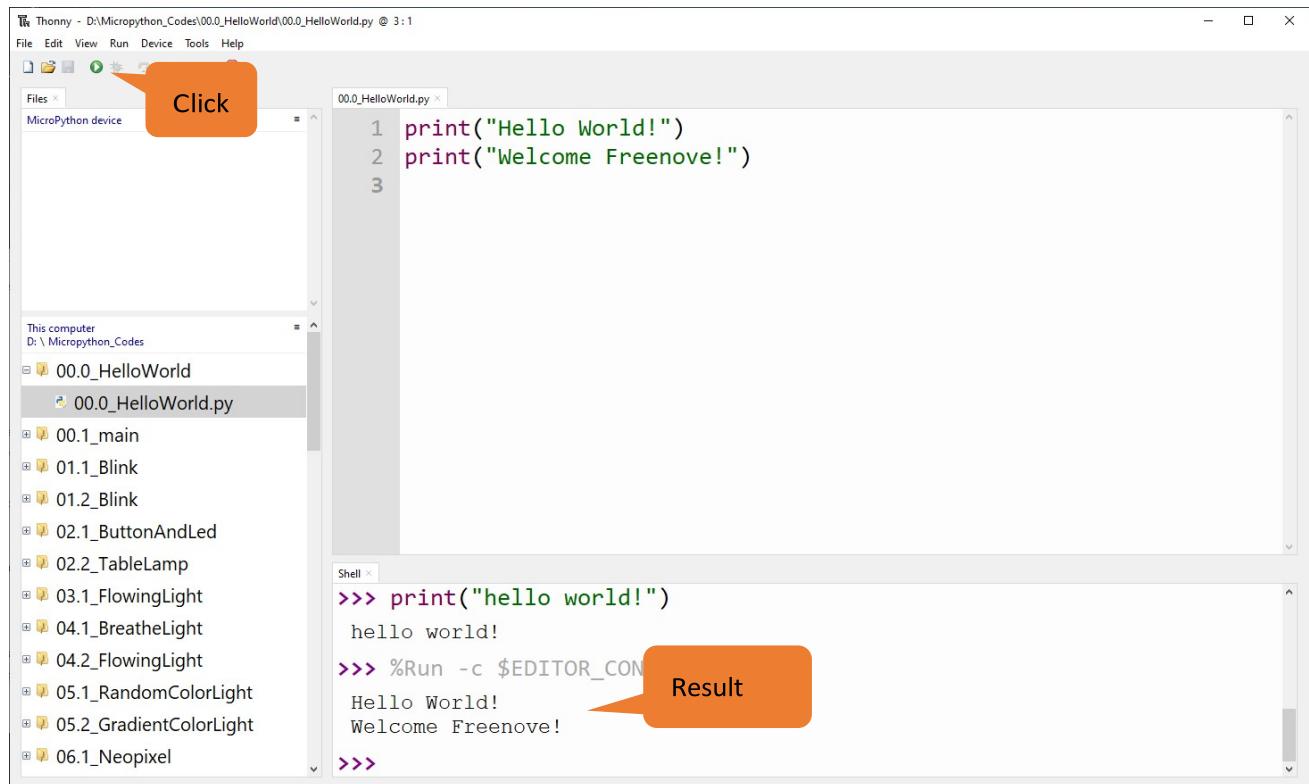
2. On the newly pop-up window, click “This computer”.



En el nuevo cuadro de diálogo , seleccione " **00.0_HelloWorld.py** " en
“ **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/00.0_HelloWorld** ”.



Haga clic en "Ejecutar script actual" para ejecutar el programa y "Hola i Mundo !", " Bienvenido Freenove" se imprimirá en "Shell".



Salir de Correr en línea

Cuando se ejecuta en línea, haga clic en "Detener / Reiniciar back- end " en thony o prensa Ctrl + C a salir del programa .



Correr sin conexión

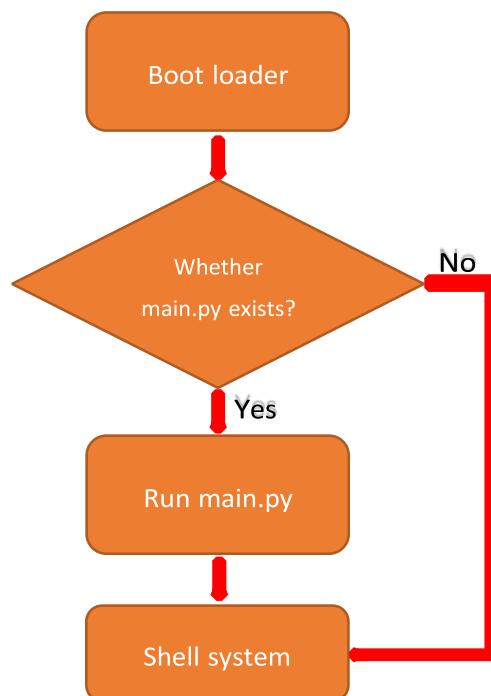
Cuando se ejecuta sin conexión, Raspberry Pi Pico no necesitar a conectar a la computadora y Thonny . Puede ejecutar los programas almacenado en main.py en el dispositivo una vez encendido .

Ventaja : puede ejecutar programas cuando encendido sin _ conectado a la computadora y Thonny .

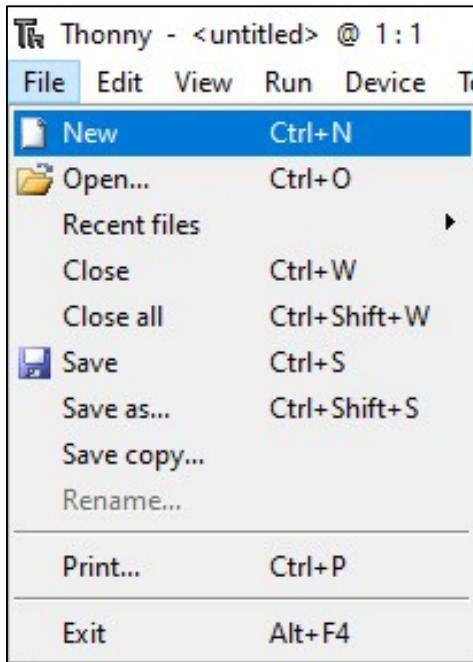
Desventaja : El programa se detendrá automáticamente cuando ocurre un error o Raspberry Pi Pico no tiene energía . Código no se puede cambiar fácilmente _

Operación

Una vez encendido , Raspberry Pi Pico comprobar automáticamente si allá es main.py existente en el dispositivo _ Si allá es __ ejecuta los programas en main.py y luego ingresar caparazón dominio sistema _ (Si tú quiero el código para ejecutar sin conexión , puede guardar como main.py); Si main.py no lo hace existe , es voluntad ingresar caparazón dominio sistema directamente _



1. Click "File" → "New" to create and write codes.



2. Enter codes in the newly opened file. Here we use codes of "01.1_Blink.py" as an example.

The screenshot shows the Thonny IDE interface with a Python script named '01.1_Blink' open in the code editor. The script contains the following code:

```

from machine import Pin
import time

led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5)
        led.value(0) # Set led turn off
        time.sleep(0.5)
except:
    pass

```

In the bottom right corner, there is a terminal window labeled 'Shell' showing the output of running the script:

```

>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

3. Haga clic en " Guardar " en la barra de menú . Puedes guardar los códigos . o a tu computadora o a Raspberry Pi Pico.

```

from machine import Pin
import time

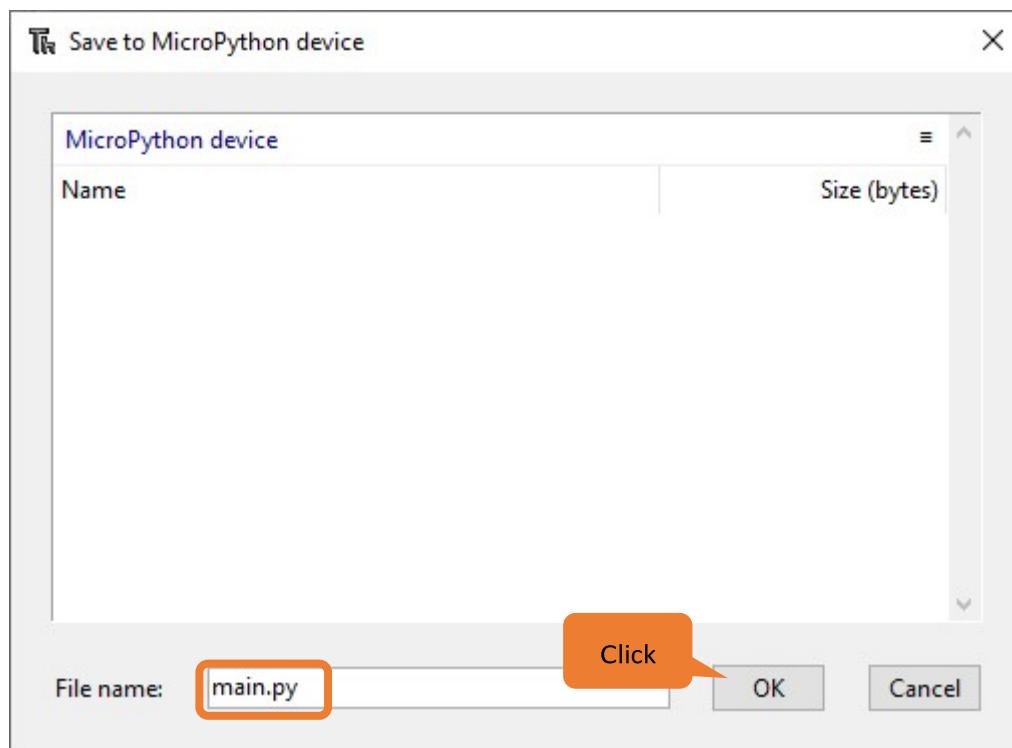
led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5)
        led.value(0) # Set led turn off
        time.sleep(0.5)
except:
    pass

```

Shell <>> print("hello world!")
hello world!
>>> %Run -c \$EDITOR_CONTENT
Hello World!
Welcome Freenove!

4. Seleccione " MicroPython dispositivo ", ingrese "main.py" en la nueva ventana emergente y haga clic en "Aceptar".



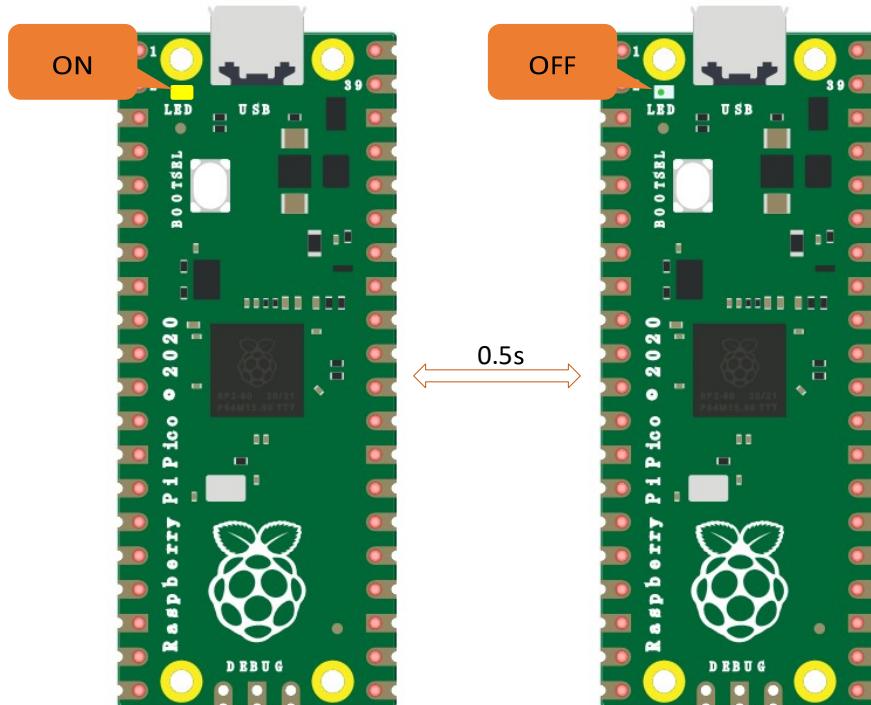
5. Puedes ver _ que codigos tener estado subido a Raspberry Pi Pico.

```

Thonny - MicroPython device :: /main.py @ 15:1
File Edit View Run Device Tools Help
[ Files ] [ main.py ]
MicroPython device
main.py
1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1) # Set led turn on
9         time.sleep(0.5) # Sleep 0.5s
10        led.value(0) # Set led turn off
11        time.sleep(0.5) # Sleep 0.5s
12    except:
13        pass
14
15
Shell <
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

6. Desconecte el cable USB Raspberry Pi Pico y luego reconectar , el LED en Raspberry Pi Pico se parpadear repetidamente _



Salir Sin conexión Ejecutar

Conecte Raspberry Pi Pico a la computadora, haga clic en "detener/ reiniciar back- end " en thony a termine de ejecutarse sin conexión.

The screenshot shows the Thonny IDE interface. A red box highlights the 'Files' tab on the left, which lists several projects under 'This computer'. An orange callout bubble points to the 'Stop' button in the top toolbar, labeled 'Click'. The main code editor window contains a script named 'main.py' with the following code:

```

from machine import Pin
import time

led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

```

The 'Shell' tab at the bottom shows the output of running the script:

```

>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

Si eso no trabaja , por favor hacer clic en "detener/ reiniciar backend " para más veces o volver a conectar Pico.

The screenshot shows the Thonny IDE interface. A red box highlights the 'Files' tab on the left, listing various projects. An orange callout bubble points to the 'Stop' button in the top toolbar, labeled 'Computer cannot read Pico internal files.' The 'Shell' tab at the bottom shows the output of running the script:

```

>>> print("Hello,world!")
Hello,world!
>>> %Run -c $EDITOR_CONTENT
LCD API v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> D<thonny>('': {'19.1_IIC_LCD1602.py': {'size': 428, 'kind': 'file', 'time': 2556144116}, 'main.py': {'size': 617, 'kind': 'file', 'time': 2556144113}, 'I2C_LCD.py': {'size': 3160, 'kind': 'file', 'time': 2556144118}, 'LCD_API.py': {'size': 6725, 'kind': 'file', 'time': 2556144121}})</thonny>

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> D<thonny>('': {'19.1_IIC_LCD1602.py': {'size': 428, 'kind': 'file', 'time': 2556144116}, 'main.py': {'size': 617, 'kind': 'file', 'time': 2556144113}, 'I2C_LCD.py': {'size': 3160, 'kind': 'file', 'time': 2556144118}, 'LCD_API.py': {'size': 6725, 'kind': 'file', 'time': 2556144121}})</thonny>

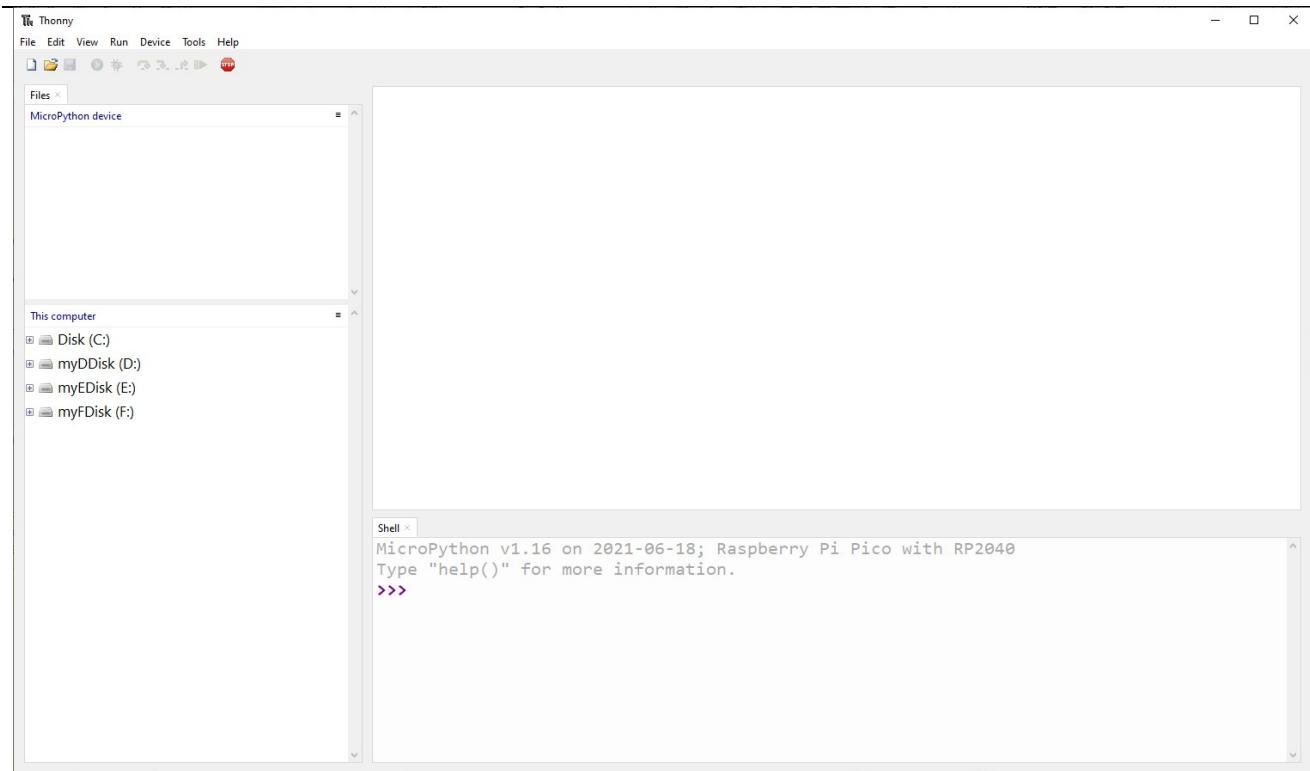
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> D<thonny>('': {'19.1_IIC_LCD1602.py': {'size': 428, 'kind': 'file', 'time': 2556144116}, 'main.py': {'size': 617, 'kind': 'file', 'time': 2556144113}, 'I2C_LCD.py': {'size': 3160, 'kind': 'file', 'time': 2556144118}, 'LCD_API.py': {'size': 6725, 'kind': 'file', 'time': 2556144121}})</thonny>

```

An orange callout bubble points to the shell output, labeled 'Shell prints Pico internal file description and other information.'

Nosotros proporcionamos un archivo main.py para ejecutar sin conexión. el código adicional a main.py es un bootstrap que ejecuta el usuario archivo de código . Todos tú necesitar hacer es _ sube el proyecto fuera de línea archivo de código (. py) al dispositivo Raspberry Pi Pico .

1. Mueva la carpeta del programa " **Freenove _Basic_Starter_Kit_for_Raspberry_Pi_Pico / Python_Codes** " al disco (D) por adelantado con la ruta de “ **D:/ Micropython_Codes** ”. Abrir " Thonny ".



2. Expanda "00.1_main" en " Micropython_Codes " en el directorio del disco (D) y haga doble clic en main.py, que es proporcionado por nosotros a habilitar programas en “ MicroPython dispositivo ” para ejecutar sin conexión.

```

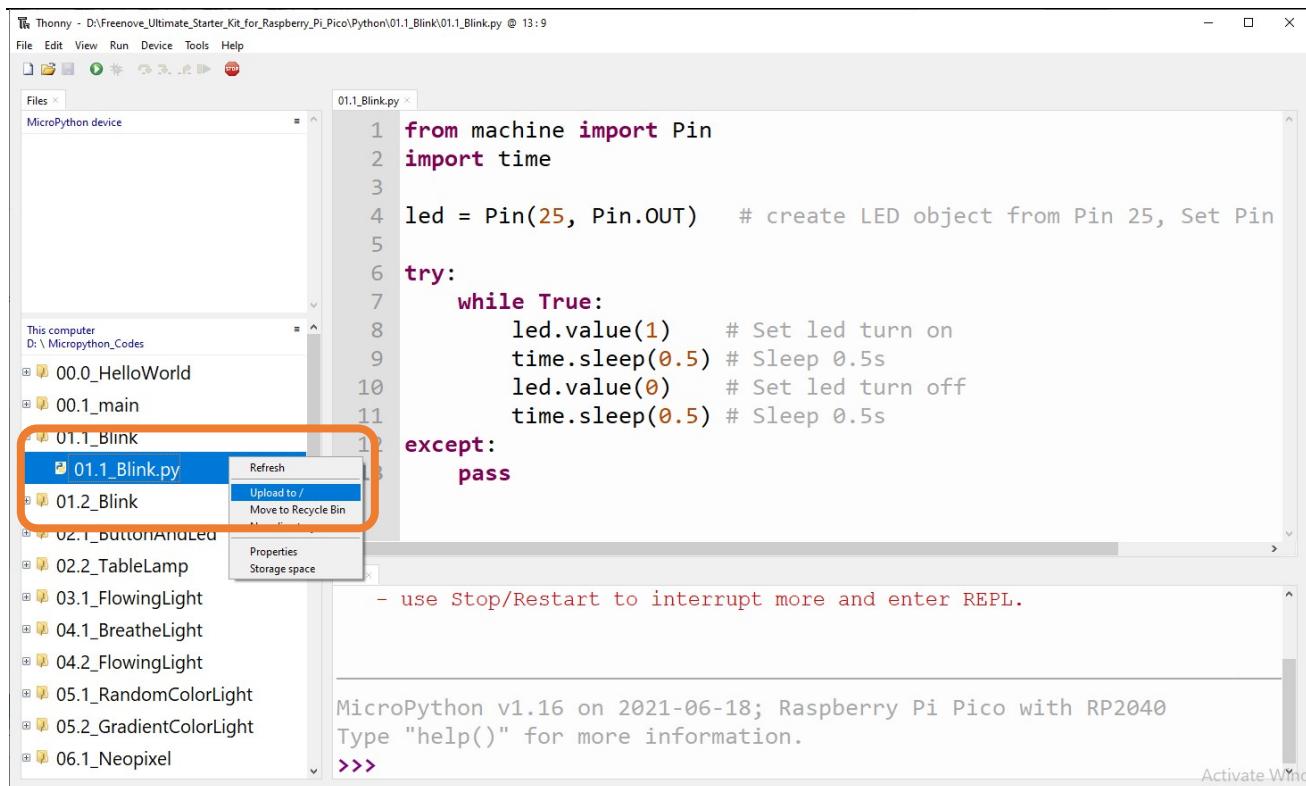
#!/opt/bin/lv_micropython
import uos as os
import uerrno as errno
iter = os.ilistdir()
IS_DIR = 0x4000
IS_REGULAR = 0x8000
index = 0

try:
    while True:
        entry = next(iter)
        filename = entry[0]
        file_type = entry[1]
        if filename == 'main.py':
            continue
        ->
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

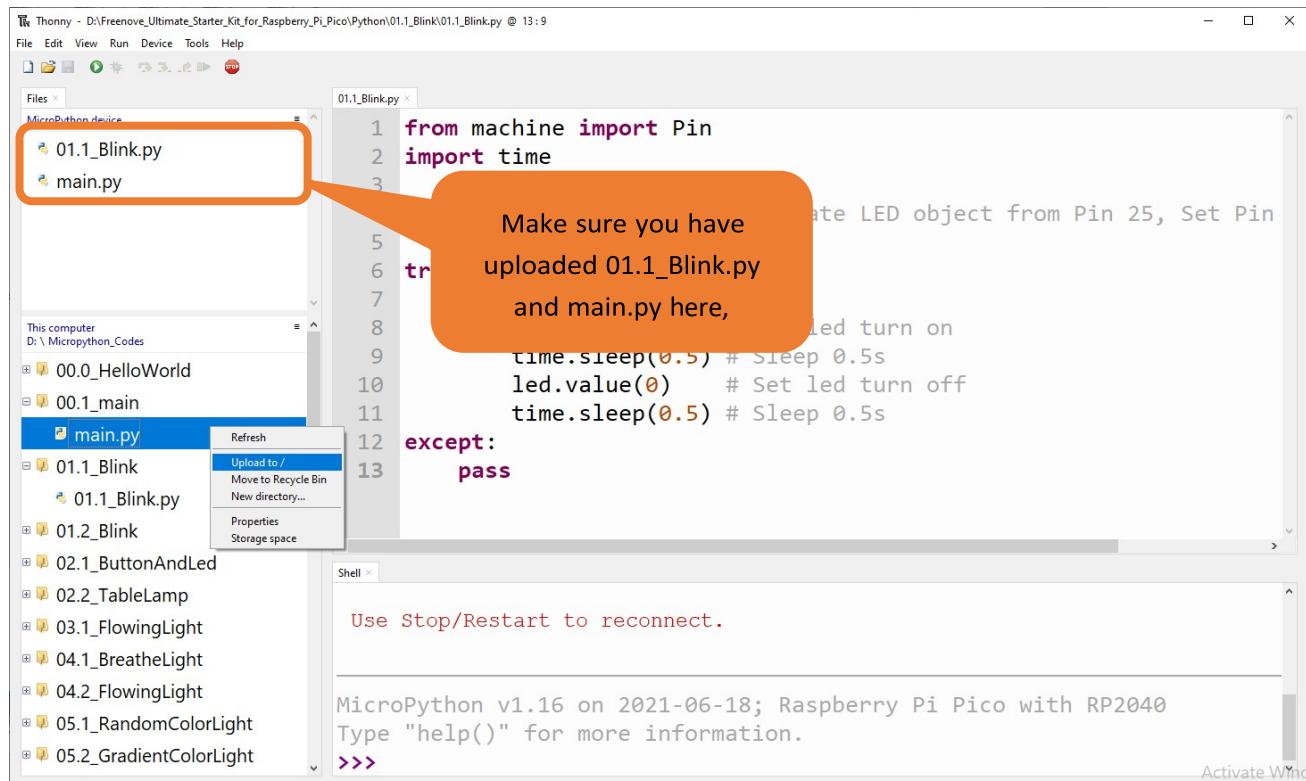
```

Aquí usamos los casos 00.1 y 01.1 como demostración . El LED en Raspberry Pi Pico es usado para mostrar el resultado , que utiliza el pin GP25. Si tú tener archivo 01.1_Blink.py modificado , usted necesitar a cambio eso en consecuencia .

Como se muestra a continuación ilustración , haga clic con el botón derecho en el archivo 01.1_Blink.py y seleccione " Cargar a /" a subir código a Raspberry Pi Pico.



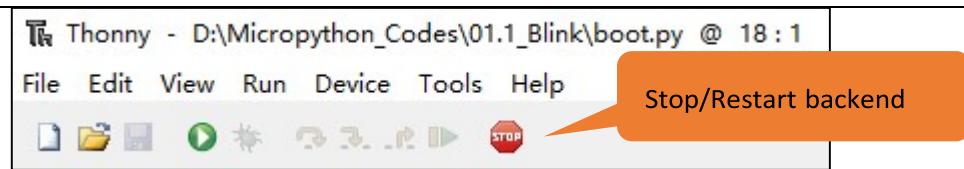
Upload main.py in the same way.



Desconecte el cable USB Raspberry Pi Pico y vuelva a conectar , el LED en pico parpadear repetidamente _

Nota:

Los códigos aquí se ejecutan sin conexión. Si tú desear para dejar de ejecutarse sin conexión e ingresar a Shell, simplemente haga clic en "Detener" en Thonny .

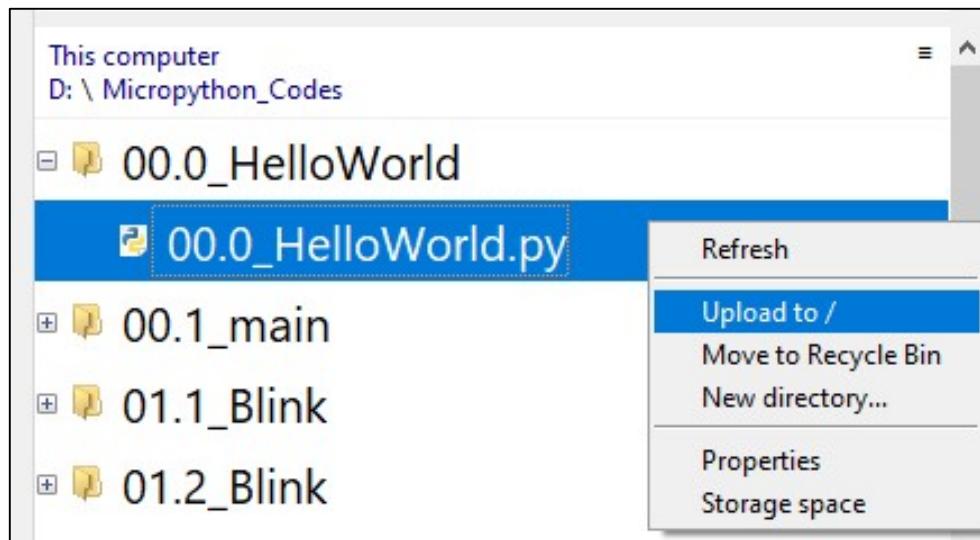


Capítulo 0 Conseguir Listo (Importante)

0.6 delgado Común Operación

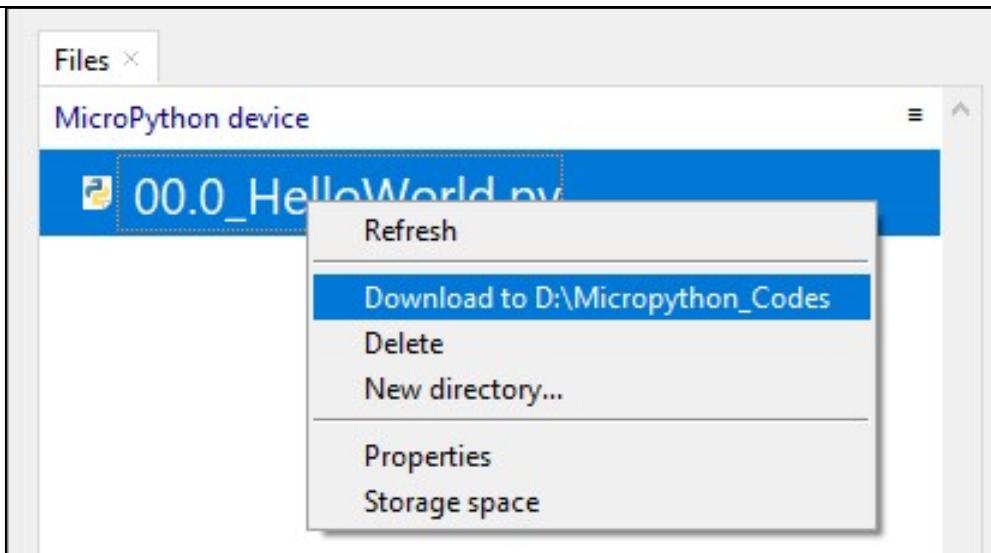
Cargando Código a Raspberry Pi Pico

Seleccione "00.0_HelloWorld.py" en "00.0_HelloWorld", haga clic con el botón derecho su mouse y seleccione "Subir a /" a subir código a Raspberry Pi Pico raíz directorio _



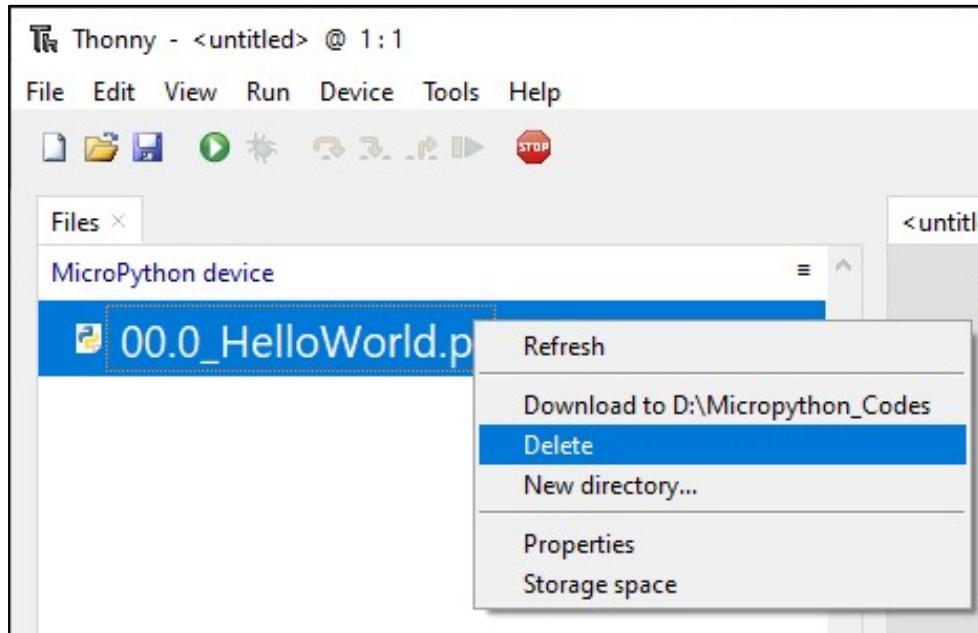
Downloading Code to Computer

Seleccione "00.0_HelloWorld.py" en " MicroPython dispositivo ", haga clic con el botón derecho a seleccione " Descargar a ..." a descargar el código a tu computadora



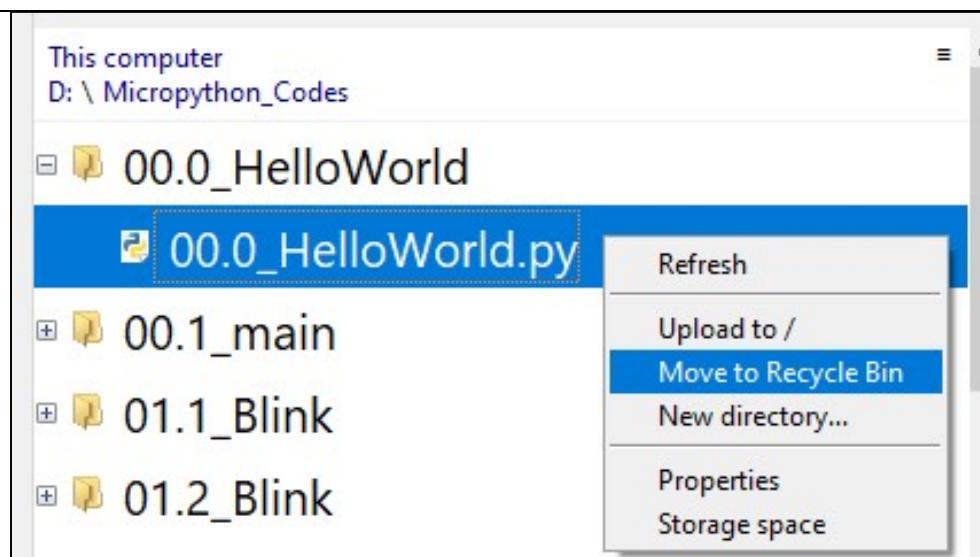
Eliminación de archivos de Raspberry Pi Pico Raíz Directorio

Seleccione "00.0_HelloWorld.py" en " MicroPython dispositivo ", haga clic con el botón derecho y seleccione " Borrar " para _ elimine "00.0_HelloWorld.py" de Raspberry Pi Pico's raíz directorio _



Eliminación de archivos de su directorio de computadoras

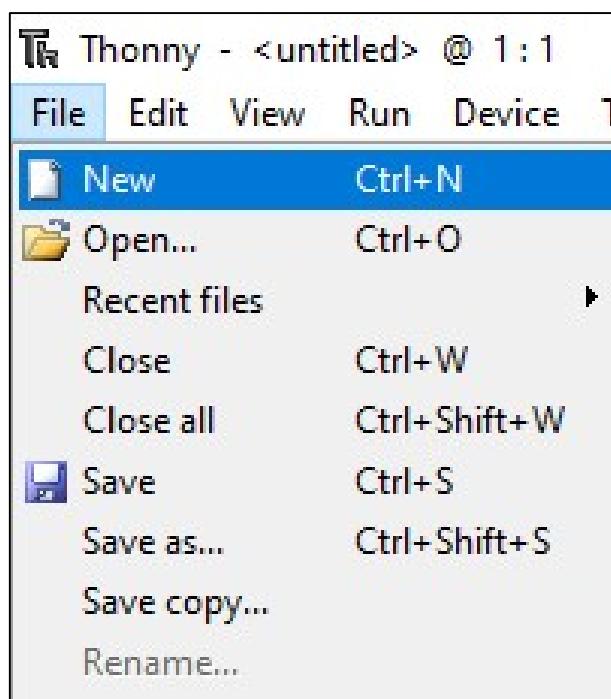
Seleccione "00.0_HelloWorld.py" en "00.0_HelloWorld", haga clic con el botón derecho y seleccione " Mover _ a Reciclar papelera " a Eliminar eso de "00.0_HolaMundo".



Capítulo 0 Conseguir Listo (Importante)

Crear y guardar el código

Click “File”→“New” to create and write codes.



Enter codes in the newly opened file. Here we use codes of “01.1_Blink.py” as an example.

The screenshot shows the Thonny IDE interface with the following components:

- File Explorer:** Shows a tree view of files under "MicroPython device". Files listed include: 00.0_HelloWorld, 01.1_main, 01.1_Blink (which is currently selected), 01.2_Blink, 02.1_ButtonAndLed, 02.2_TableLamp, 03.1_FlowingLight, 04.1_BreatheLight, 04.2_FlowingLight, 05.1_RandomColorLight, 05.2_GradientColorLight, 06.1_Neopixel, and 06.2_Rainbow_light.
- Code Editor:** The main window displays the code for "01.1_Blink.py":

```

1 from machine import Pin
2 import time
3
4 led = Pin(25, Pin.OUT)      # create LED object from Pin 25, Set Pin
5
6 try:
7     while True:
8         led.value(1)        # Set led turn on
9         time.sleep(0.5)     # Sleep 0.5s
10        led.value(0)       # Set led turn off
11        time.sleep(0.5)    # Sleep 0.5s
12 except:
13     pass

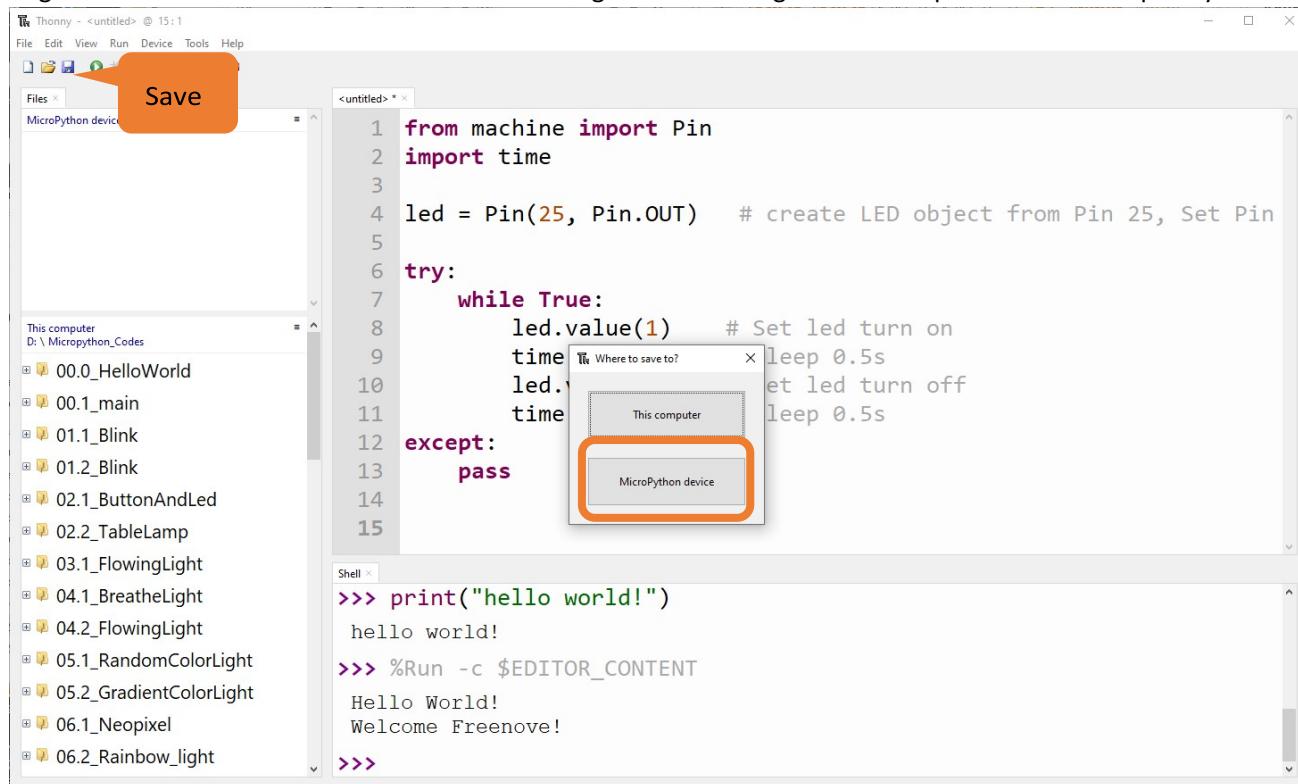
```
- Shell:** A terminal window at the bottom shows the output of running the code:

```

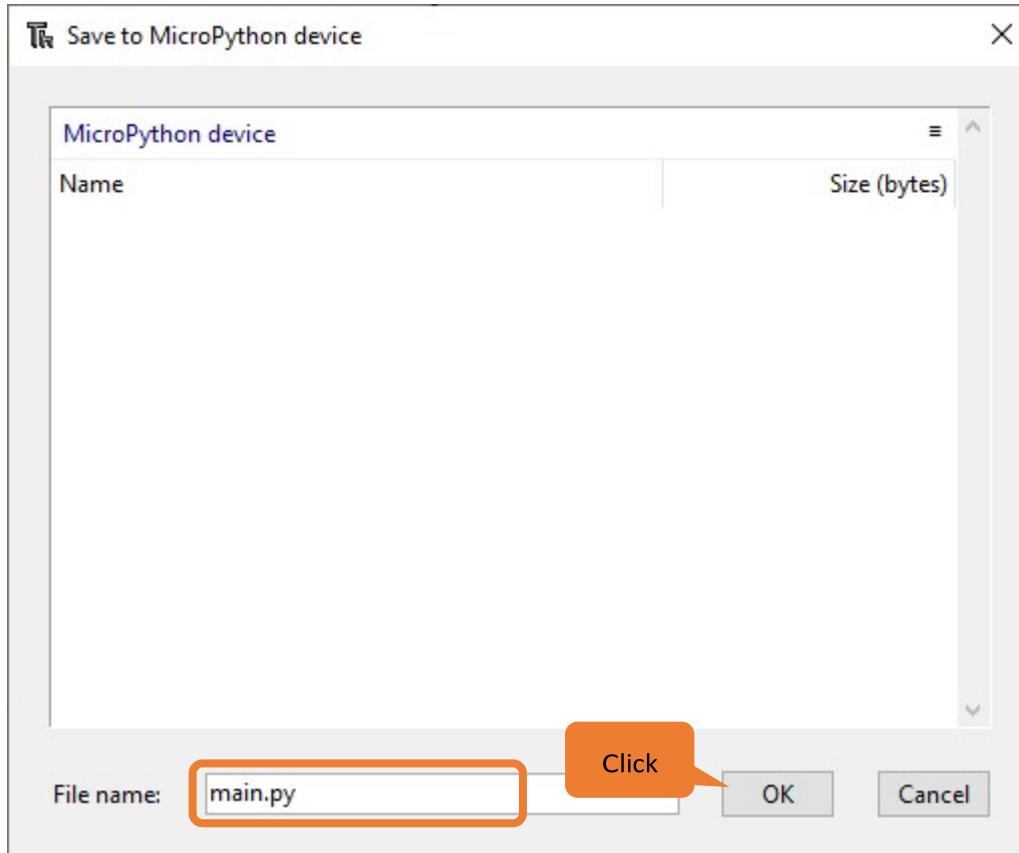
>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

Haga clic en "Guardar" en la barra de menú. Puede guardar los códigos en su computadora o en Raspberry Pi Pico.

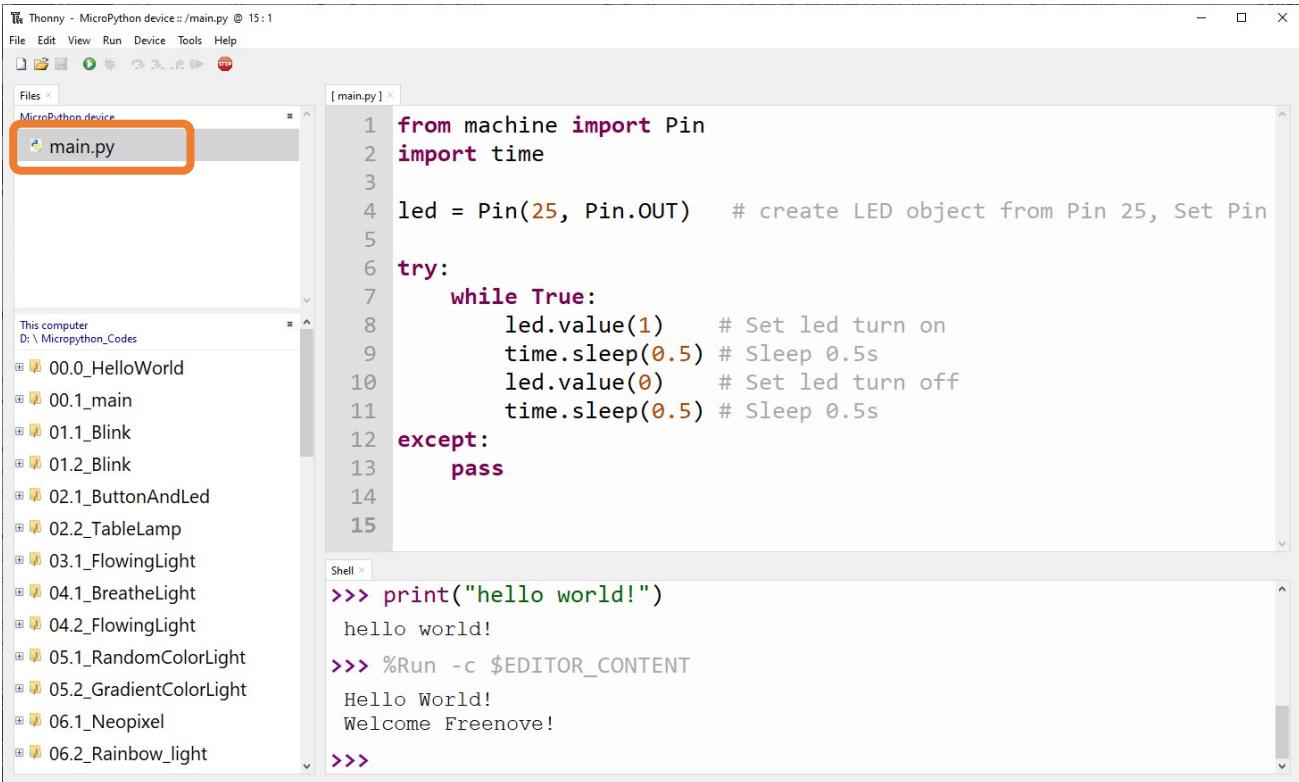


Seleccione "Dispositivo MicroPython", ingrese "main.py" en la nueva ventana emergente y haga clic en "Aceptar".



Capítulo 0 Preparándose (Importante)

Puede ver que los códigos se han cargado en Raspberry Pi Pico.



The screenshot shows the Thonny IDE interface. On the left, the 'Files' sidebar lists several Python files under 'D:\Micropython_Codes'. The 'main.py' file is selected and highlighted with an orange box. The main workspace shows the code for a blinking LED:

```

from machine import Pin
import time

led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

```

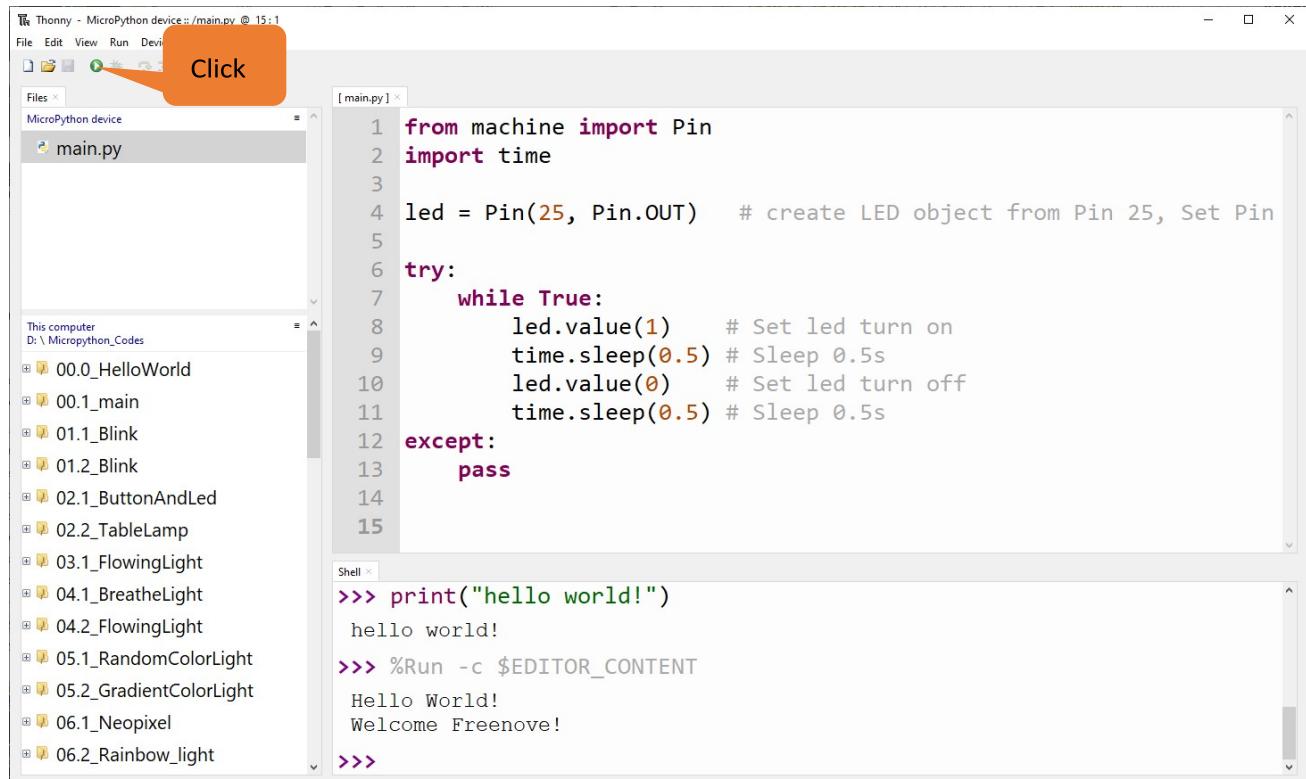
The 'Shell' tab at the bottom contains the output of the code execution:

```

>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

Click "Run" and the LED on Raspberry Pi Pico will blink periodically.



The screenshot shows the Thonny IDE interface again. The 'main.py' file is selected. An orange box highlights the 'Run' button in the toolbar. The main workspace shows the same code as before. The 'Shell' tab shows the output:

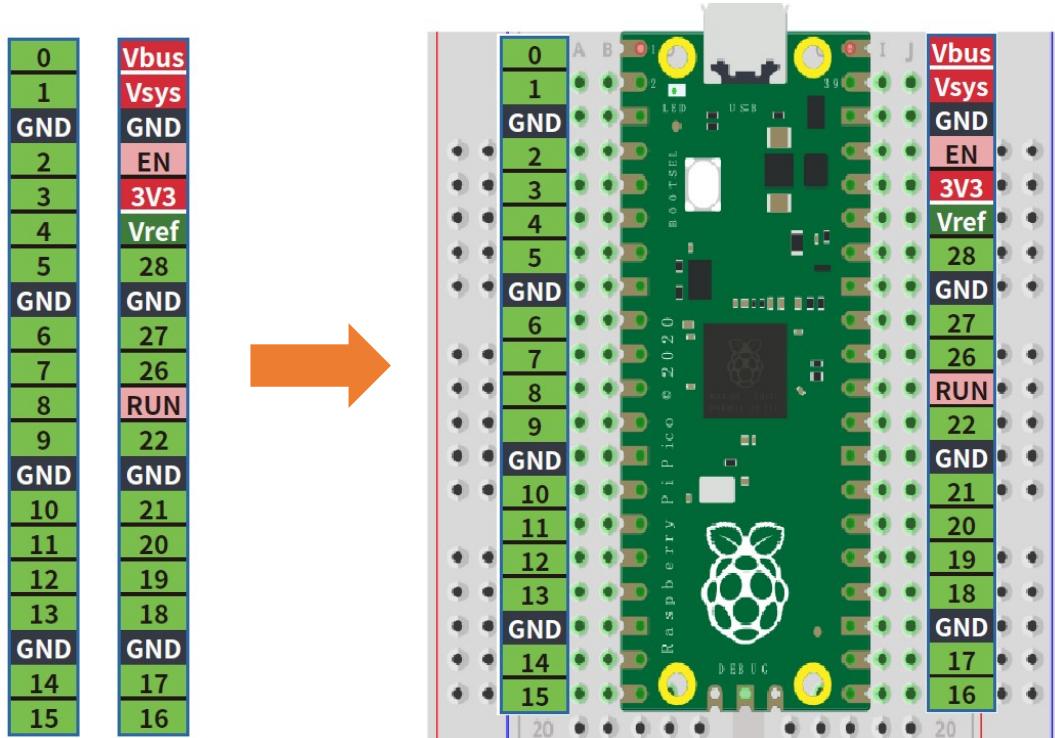
```

>>> print("hello world!")
hello world!
>>> %Run -c $EDITOR_CONTENT
Hello World!
Welcome Freenove!
>>>

```

06. Pega la pegatina en la placa de pruebas

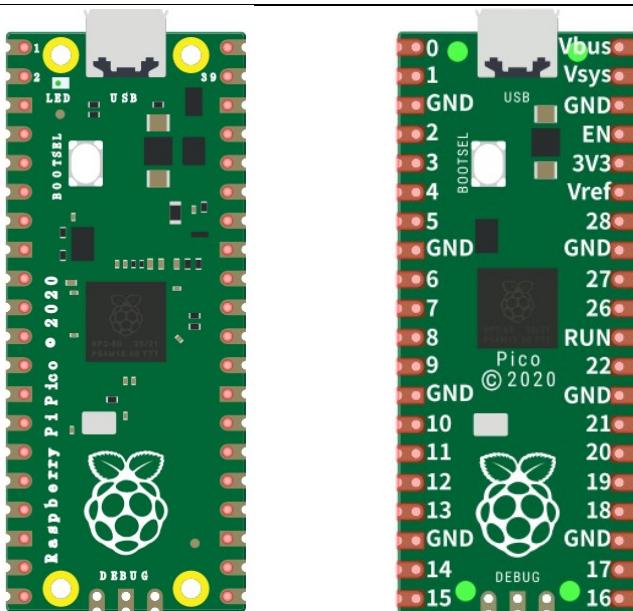
No es difícil usar el Pico. Sin embargo, oficialmente, las funciones de los pines están impresas en la parte posterior de la placa, lo que hace que su uso sea inconveniente. Para ayudar a los usuarios a terminar cada proyecto en el tutorial de forma más rápida y sencilla, proporcionamos pegatinas de las funciones de los pines de la siguiente manera:



Puede pegar la pegatina en el área en blanco de la placa de prueba como se indica arriba.

Nota: La secuencia de pines funcionales de Pico y Pico W es la misma. Por lo tanto, incluso si su kit es Pico W, también es aplicable a la etiqueta de arriba.

Para que el tutorial sea más intuitivo, hemos realizado algunos cambios en el diagrama de simulación, como se muestra a continuación. El de la izquierda es el Pico real y el de la derecha es su diagrama de simulación. Tenga en cuenta que para evitar malentendidos.



Capítulo 1 LED (Importante)

Capítulo 1 LED (Importante)

Nota: Raspberry Pi Pico y Raspberry Pi Pico W solo difieren en una función inalámbrica y son casi idénticas en otros aspectos. En este tutorial, a excepción de la función inalámbrica, otras partes usan el mapa de Raspberry Pi Pico para la demostración del tutorial.

Este capítulo es el punto de partida en el viaje para construir y explorar proyectos electrónicos de Raspberry Pi Pico. Comenzaremos con el proyecto simple "Blink".

Proyecto 1.1 Parpadeo

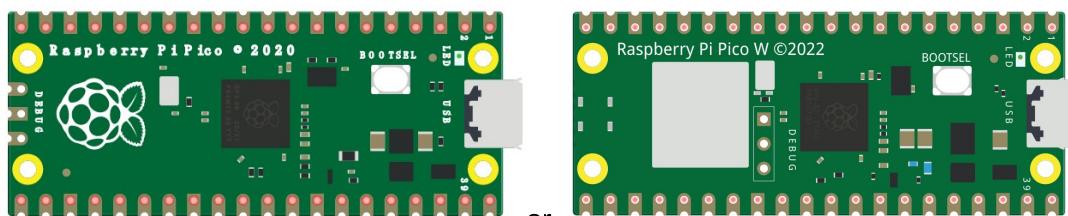
En este proyecto, usaremos Raspberry Pi Pico para controlar el parpadeo de un LED común.

Si aún no ha instalado Thonny, haga clic [aquí](#).

Si aún no ha descargado el firmware de Micropython, haga clic [aquí](#). Si aún no ha cargado el firmware de Micropython, haga clic [aquí](#).

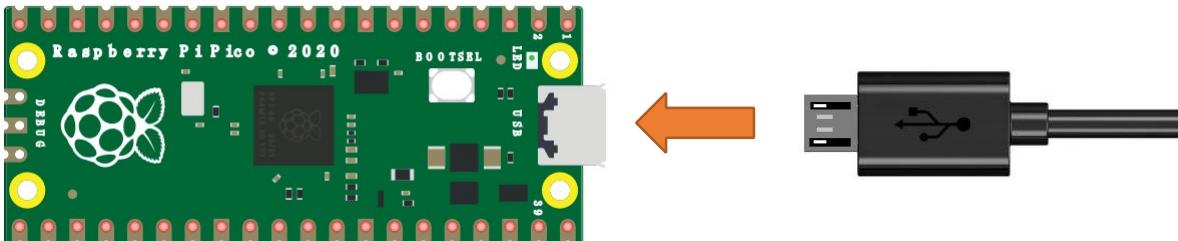
Lista de componentes

Raspberry Pi Pico (o Pico W) x1



Cable USBx1**Energía**

En este tutorial, conectamos Raspberry Pi Pico y computer con un cable USB.



Código

Los códigos utilizados en este tutorial se guardan en

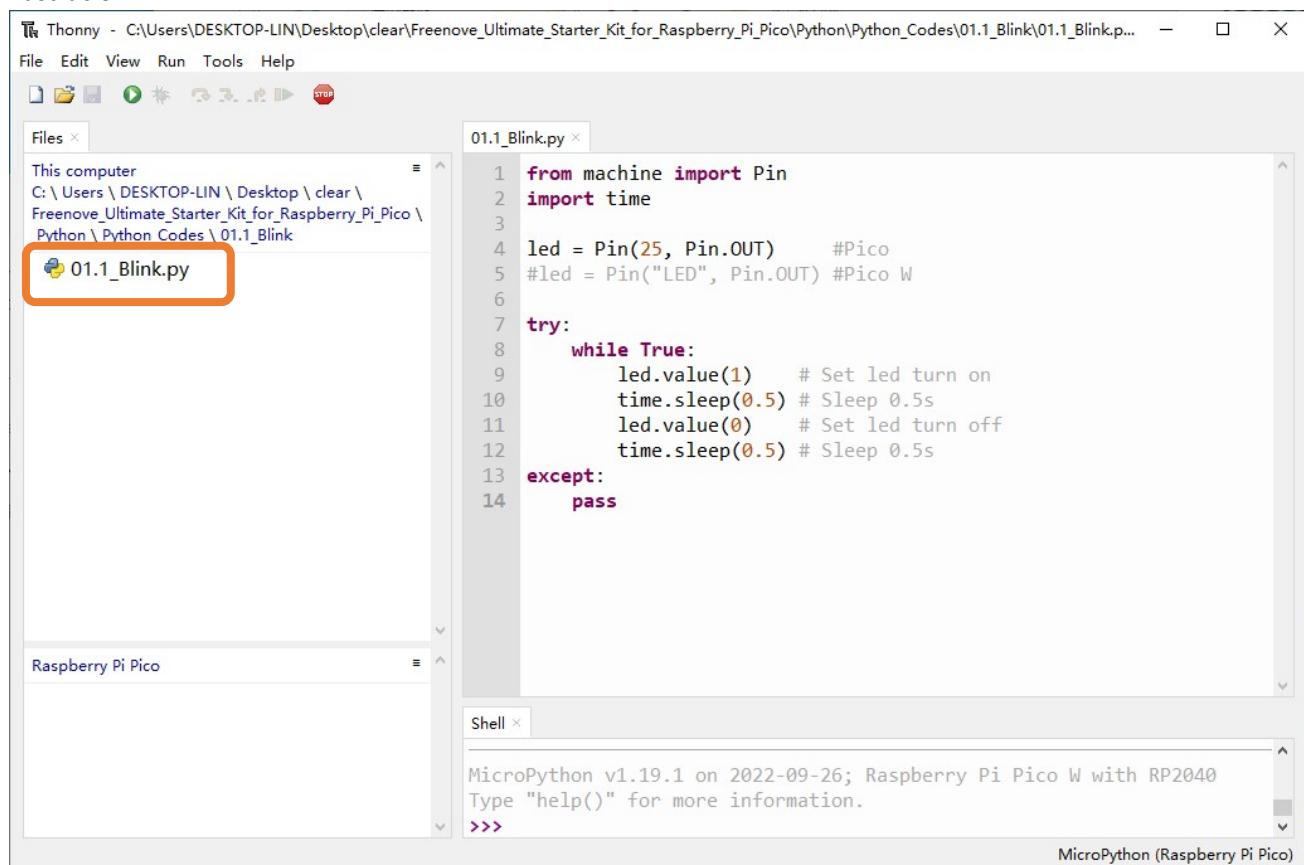
“ **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python_Codes** ”. Puede mover los códigos a cualquier ubicación. Por ejemplo, guardamos los códigos en Disk(D) con la ruta de “ **D:/Micropython_Codes** ”.

01.1_parpadeo

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes".



Expanda la carpeta "01.1_Blink" y haga doble clic en "01.1_Blink.py" para abrirla. Como se muestra en la siguiente ilustración.



Capítulo 1 LED (Importante)

Asegúrese de que Raspberry Pi Pico se haya conectado con la computadora. Haga clic en "Detener/Reiniciar backend" y luego espere a ver qué interfaz aparecerá.

```

from machine import Pin
import time

led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

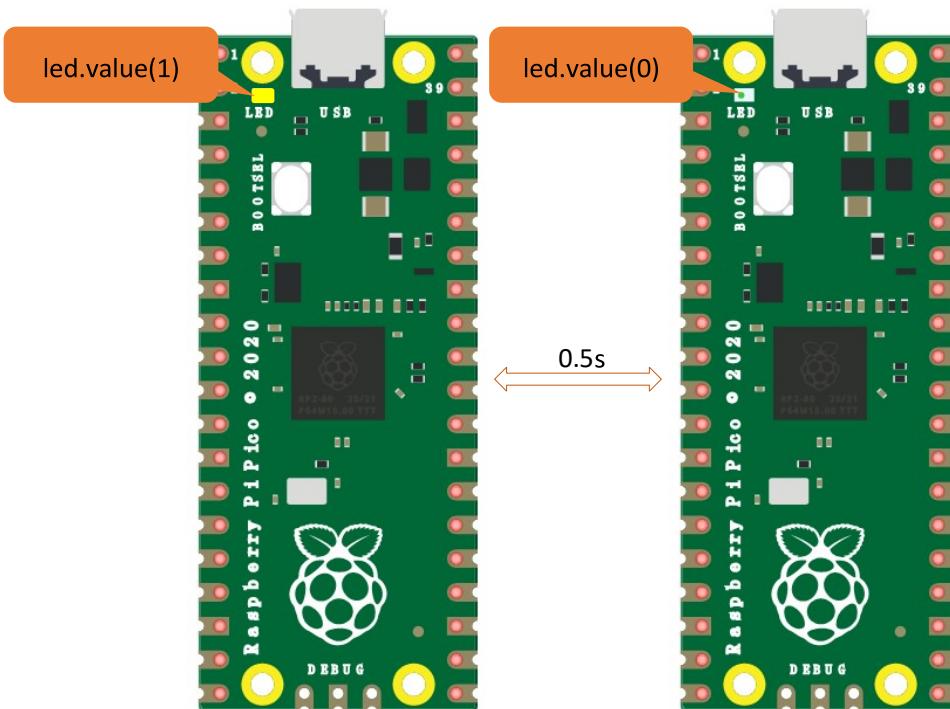
```

Use Stop/Restart to reconnect.

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico Type "help()" for more information.

>>>

Haga clic en "Ejecutar secuencia de comandos actual" que se muestra en el cuadro de arriba, el código comienza a ejecutarse y el LED en el circuito comienza a parpadear. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



Nota :

Este es el código que se [ejecuta en línea](#). Si desconecta el cable USB y vuelve a encender Raspberry Pi Pico, el LED deja de parpadear y se mostrarán los siguientes mensajes en Thonny.

The screenshot shows the Thonny IDE interface. On the left, the file tree displays various Python scripts under 'D:\MicroPython_Codes'. The script '01.1_Blink.py' is selected and shown in the main code editor. The code itself is a simple LED blink program. Below the code editor is a 'Shell' window showing MicroPython v1.16 running on a Raspberry Pi Pico with RP2040. It shows a connection loss and a message to use Stop/Restart to reconnect.

```

from machine import Pin
import time

led = Pin(25, Pin.OUT) # create LED object from Pin 25, Set Pin

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

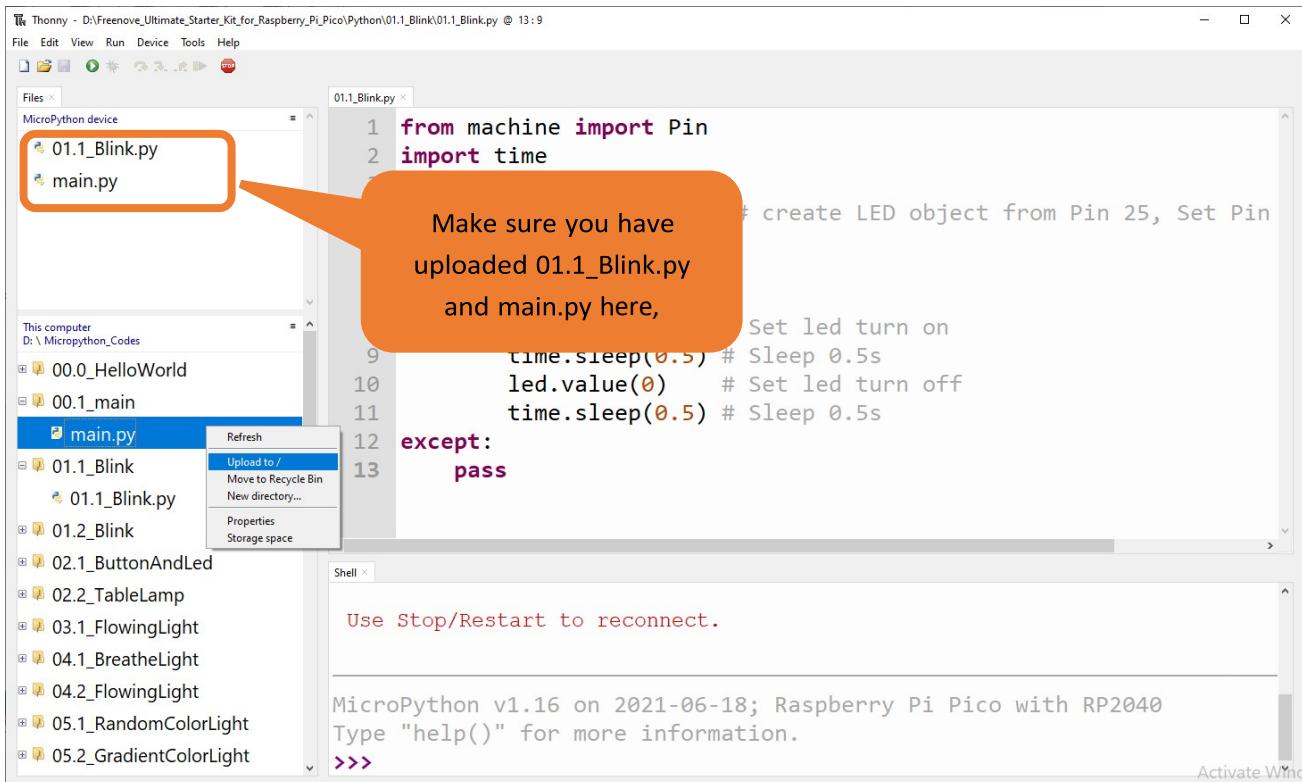
```

Subir código a Raspberry Pi Pico

Como se muestra en la siguiente ilustración, haga clic con el botón derecho en el archivo 01.1_Blink.py y seleccione "Cargar en /" para cargar el código en Raspberry Pi Pico.

This screenshot shows the same Thonny IDE setup as above, but with a right-click context menu open over the '01.1_Blink.py' file in the file tree. The 'Upload to /' option is highlighted in blue, indicating it is the selected action. The rest of the interface and code are identical to the previous screenshot.

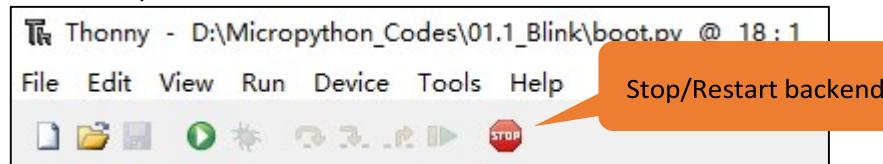
Upload main.py in the same way.



Desconecte el cable USB de Raspberry Pi Pico y vuelva a conectarlo, el LED en Pico parpadeará repetidamente.

Nota:

Los códigos aquí se ejecutan sin conexión. Si desea dejar de ejecutar sin conexión e ingresar a Shell, simplemente haga clic en "Detener" en Thonny.



Si usted tiene alguna inquietud, contáctenos a través de: support@freenove.com

El siguiente es el

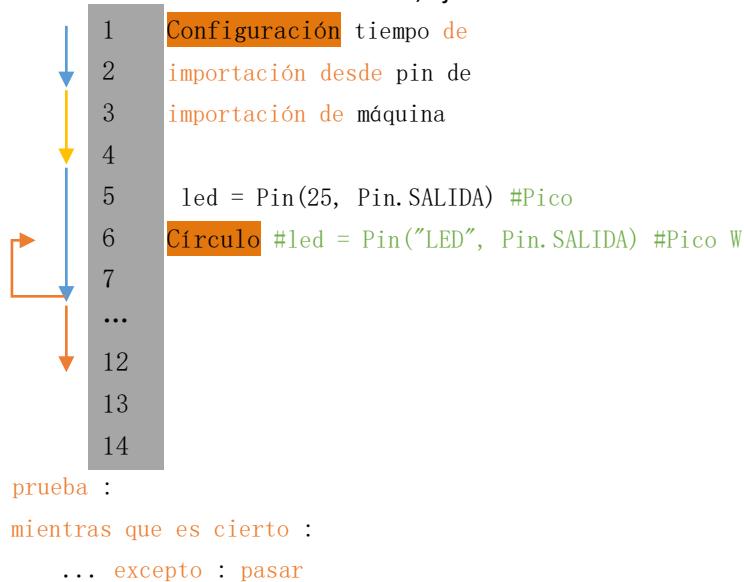
código del programa:

```

1 tiempo de importación
2 desde pin de
3 importación de máquina
4 led = Pin(25, Pin.SALIDA) #Pico
5 #led = Pin("LED", Pin.SALIDA) #Pico W
6 prueba :
7 mientras que es
8 cierto :
9 led.valor(1) #Establecer el tiempo de
10 encendido del led.sleep(0.5) #Sleep 0.5s
11 led.value(0) #Establecer el tiempo de
12 apagado del led.sleep(0.5) #Sleep 0.5s
13 excepto : pasar
14

```

Cada vez que se abre un nuevo archivo, el programa se ejecutará de arriba a abajo. Cuando encuentre una construcción de bucle, ejecutará la declaración de bucle de acuerdo con la condición del bucle.



La función Print() se utiliza para imprimir datos en la Terminal. Se puede ejecutar en Terminal directamente o escribirse en un archivo de Python y ejecutarse ejecutando el archivo.

```
imprimir ( "¡Hola mundo!" )
```

Cada vez que utilice las funciones de Raspberry Pi Pico, debe importar los módulos correspondientes a esas funciones: importar el módulo de tiempo y el módulo Pin del módulo de la máquina.

```

1 tiempo de importación
2 desde pin de importación de máquina

```

Configure GP25 de Raspberry Pi Pico en modo de salida y asígnelo a un objeto llamado "led".

Configure "LED" de Raspberry Pi Pico W en modo de salida y asígnelo a un objeto llamado "led".

```

4 led = Pin(25, Pin.SALIDA) #Pico
5 #led = Pin("LED", Pin.SALIDA) #Pico W

```

Significa que a partir de ahora, el LED que representa GP25 está en modo de salida. Establezca el valor de LED en 1 y GP25 emitirá un nivel alto.

```
8 led.valor(1) #Establecer encendido de led
```

Establezca el valor del LED en 0 y el pin LED emitirá un nivel bajo.

```
10 led.valor(0) #Establecer encendido de led
```

Ejecuta códigos en un bucle while.

```
7 mientras que es cierto :  
...  
13 ...
```

Coloque las declaraciones que pueden causar un error en el bloque "intentar" y las declaraciones de ejecución cuando se produce un error en el bloque "excepto". En general, cuando el programa ejecuta declaraciones, ejecutará aquellas en el bloque "intentar". Sin embargo, cuando ocurre un error en Raspberry Pi Pico debido a alguna interferencia u otras razones, ejecutará declaraciones en el bloque "excepto".

"Pass" es una declaración vacía. Cuando se ejecuta, no pasa nada. Es útil como marcador de posición para mejorar la apariencia de la estructura de un programa.

```
6 prueba :  
...  
12 ...  
13 excepto :  
    pasar
```

El comentario de una sola línea de Micropython comienza con un "#" y continúa hasta el final de la línea. Los comentarios nos ayudan a entender el código. Cuando los programas se están ejecutando, Thonny omitirá los comentarios.

```
8 #Establecer encendido de led
```

MicroPython usa sangrías para distinguir diferentes bloques de código en lugar de llaves. El número de sangrías se puede cambiar, pero debe ser consistente a lo largo de un bloque. Si la sangría del mismo bloque de código es inconsistente, provocará errores cuando se ejecute el programa.

```
7 mientras que es cierto :  
8     led.valor(1) #Establecer el tiempo de encendido  
9     del led.sleep(0.5) #Sleep 0.5s led.value(0)  
10    #Establecer el tiempo de apagado del  
11    led.sleep(0.5) #Sleep 0.5s
```

Cómo importar archivos de Python

Si importa el módulo directamente, debe indicar el módulo al que pertenece la función o el atributo al usar la función o el atributo (constante, variable) en el módulo. El formato debe ser: <nombre del módulo>.<función o atributo>, de lo contrario se producirá un error.

```
import random  
num = random.randint(1, 100)  
print(num)
```

Si solo desea importar una determinada función o atributo en el módulo, use la instrucción from...import. El formato es el siguiente

```
from random import randint  
num = randint(1, 100)  
print(num)
```

Cuando utilice la instrucción "from ... import" para importar la función, para evitar conflictos y facilitar la comprensión, puede utilizar la instrucción "as" para cambiar el nombre de la función importada, de la siguiente manera

```
from random import randint as rand
num = rand(1, 100)
print(num)
```

Referencia

máquina de clase	<p>Antes de cada uso del módulo de la máquina, agregue la declaración "importar máquina" en la parte superior del archivo python.</p> <p>machine.freq(freq_val): Cuando no se especifica "freq_val", es para volver a la frecuencia actual de la CPU; De lo contrario, es para establecer la frecuencia actual de la CPU. valor_frecuencia : 125000000Hz (125MHz).</p> <p>machine.reset() : Una función de reinicio. Cuando se llama, el programa se reiniciará. machine.unique_id() : Obtiene la dirección MAC del dispositivo.</p> <p>machine.idle() : Desactiva cualquier función no utilizada temporalmente en el chip y su reloj, lo que es útil para reducir el consumo de energía en cualquier momento durante períodos cortos o largos.</p> <p>machine.disable_irq() : deshabilita las solicitudes de interrupción y devuelve el estado IRQ anterior. La función disabled_irq () y la función enable_irq () deben usarse juntas; De lo contrario, la máquina se bloqueará y se reiniciará.</p> <p>machine.enable_irq(state) : Para volver a habilitar las solicitudes de interrupción. El estado del parámetro debe ser el valor que se devolvió desde la llamada más reciente a la función disabled_irq().</p> <p>machine.time_pulse_us(pin, pulse_level, timeout_us=1000000) :</p> <p>Prueba la duración del nivel de pulso externo en el pin dado y devuelve la duración del nivel de pulso externo en microsegundos. Cuando el nivel de pulso = 1, prueba la duración del nivel alto; Cuando el nivel de pulso = 0, prueba la duración del nivel bajo.</p> <p>Si el nivel de configuración no es consistente con el nivel de pulso actual, esperará hasta que sean consistentes y luego comenzará a cronometrar. Si el nivel establecido es consistente con el nivel de pulso actual, comenzará a contar el tiempo inmediatamente.</p> <p>Cuando el nivel del pin es opuesto al nivel establecido, esperará el tiempo de espera y devolverá "-2". Cuando el nivel del pin y el nivel establecido es el mismo, también esperará el tiempo de espera pero devolverá "-1". timeout_us es la duración del tiempo de espera.</p> <p>Para obtener más información sobre la clase y la función, consulte:</p> <p>https://docs.micropython.org/en/latest/rp2/quickref.html</p>
Hora de clase	

Antes de cada uso del módulo de **tiempo**, agregue la declaración " tiempo de importación " en la parte superior del archivo python **time.sleep(sec)** : duerme durante la cantidad de segundos dada.

sec : este argumento debe ser un int o un float.

time.sleep_ms(ms) : Duerme por el número dado de milisegundos, ms debe ser un int.

time.sleep_us(us) : Duerme por el número dado de microsegundos, us debería ser un int. **time.time()** : Obtiene la marca de tiempo de la CPU, con el segundo como unidad.

time.ticks_ms() : Devuelve el valor del contador de milisegundos incremental, que vuelve a contar después de algunos valores.

time.ticks_us() : Devuelve microsegundos.

time.ticks_cpu() : Similar a ticks_ms() y ticks_us(), pero es más preciso (reloj de retorno de la CPU).

time.ticks_add(ticks, delta) : Obtiene la marca de tiempo después del desplazamiento.

ticks : ticks_ms(), ticks_us(), ticks_cpu().

delta : Delta puede ser un número entero arbitrario o una expresión numérica.

time.ticks_diff(old_t, new_t) : Calcula el intervalo entre dos marcas de tiempo, como ticks_ms(), ticks_us() o ticks_cpu().

old_t : Hora de inicio. **new_t** :

Hora de finalización.

Pin de clase (id, modo, extracción, valor)

Antes de cada uso del módulo **Pin**, agregue la declaración " from machine import Pin " en la parte superior del archivo python. **id**: Número de pin arbitrario.

modo : Modo de pines.

Pin.IN : Modo de entrada.

Pin.OUT : Modo de salida.

Pin.OPEN_DRAIN : Modo de drenaje abierto.

Tirar : si se habilita el modo interno de subir y bajar.

Ninguno : sin resistencias pull up o pull down.

Pin.PULL_UP : Modo pull-up, con salida de alto nivel por defecto.

Pin.PULL_DOWN : modo desplegable, con salida de bajo nivel de forma predeterminada.

Valor : Estado del nivel del pin, 0/1.

Pin.init (modo, extracción) : inicializa los pines.

Pin.value([value]) : Obtenga o establezca el estado del nivel del pin, devuelva 0 o 1 según el nivel lógico de los pines. Sin parámetro, lee el nivel de entrada. Con el parámetro dado, es para establecer el nivel de salida.

valor : Puede ser Verdadero/Falso o 1/0.

Pin.irq(trigger, handler) : Configura un controlador de interrupción para que se llame cuando el nivel del pin cumpla una condición.

disparador :

Pin.IRQ_FALLING : interrupción en el flanco descendente.

Pin.IRQ_RISING : interrupción en flanco ascendente.

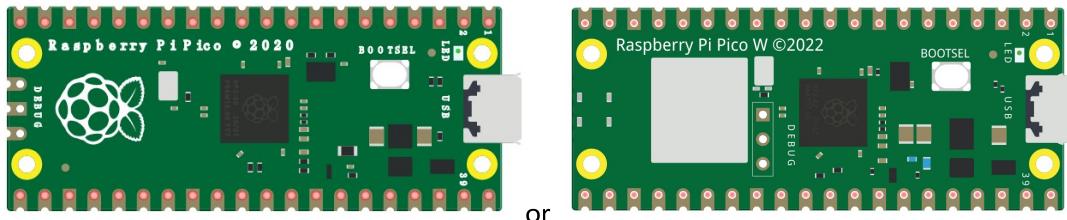
Controlador : función de devolución de llamada.

Proyecto 1.2 Parpadeo

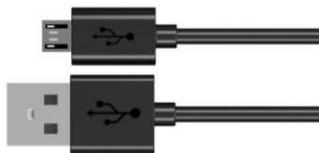
En este proyecto, usaremos Raspberry Pi Pico para controlar el parpadeo de un LED común.

Lista de componentes

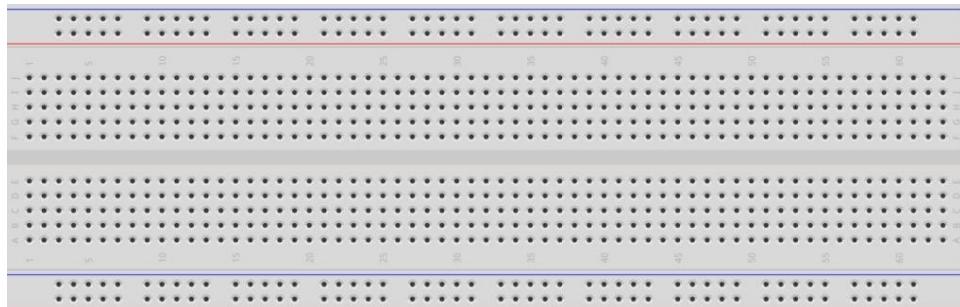
Raspberry Pi Pico (o Pico W) x1



Cable USBx1



Protopboard x1



LED x1



Resistencia 220Ω x1



Puente



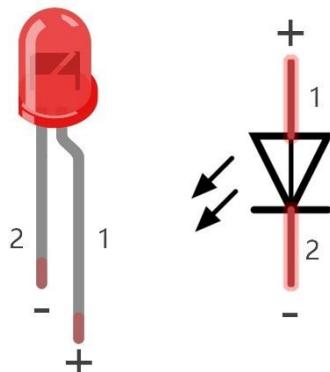
Conocimiento de componentes

DIRIGIÓN

Un LED es un tipo de diodo. Todos los diodos solo funcionan si la corriente fluye en la dirección correcta y tienen dos polos. Un LED solo funcionará (se encenderá) si el pin más largo (+) del LED está conectado a la salida positiva

¿Cualquier duda? support@freenove.com

de una fuente de alimentación y el pin más corto está conectado al negativo (-). La salida negativa también se conoce como Tierra (GND). Este tipo de componente se conoce como "Polar" (piense en una calle de sentido único). Todos los diodos comunes de 2 conductores son iguales a este respecto. Los diodos funcionan solo si el voltaje de su electrodo positivo es mayor que su electrodo negativo y existe un rango estrecho de voltaje operativo para la mayoría de los diodos comunes de 1,9 y 3,4 V. Si usa mucho más de 3.3V, el LED se dañará y se quemará.



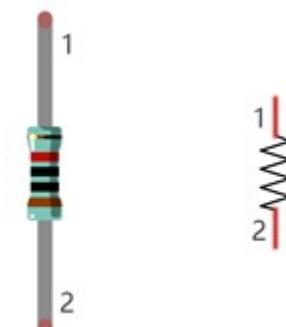
LED	Voltage	Maximum current	Recommended current
Red	1.9-2.2V	20mA	10mA
Green	2.9-3.4V	10mA	5mA
Blue	2.9-3.4V	10mA	5mA
Volt ampere characteristics conform to diode			

Nota: los LED no se pueden conectar directamente a una fuente de alimentación, lo que generalmente termina en un componente dañado. Una resistencia con un valor de resistencia específico debe conectarse en serie al LED que planea usar.

Resistor

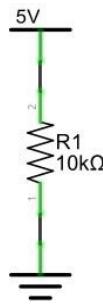
Las resistencias utilizan ohmios (Ω) como unidad de medida de su resistencia (R). $1M\Omega=1000k\Omega$, $1k\Omega=1000\Omega$.

Una resistencia es un componente eléctrico pasivo que limita o regula el flujo de corriente en un circuito electrónico. A la izquierda, vemos una representación física de una resistencia, y a la derecha está el símbolo que se usa para representar la presencia de una resistencia en un diagrama o esquema de circuito.



Las bandas de color en una resistencia son un código abreviado que se usa para identificar su valor de resistencia. Para obtener más detalles sobre los códigos de color de las resistencias, consulte el apéndice de este tutorial. Con un voltaje fijo, habrá menos salida de corriente con mayor resistencia añadida al circuito. La relación entre Corriente, Voltaje y Resistencia se puede expresar mediante esta fórmula: $I=V/R$ conocida como Ley de Ohm donde I = Corriente, V = Voltaje y R = Resistencia. Conocer los valores de cualquiera de estos dos te permite resolver el valor del tercero.

En el siguiente diagrama, la corriente a través de R1 es: $I=U/R=5V/10k\Omega=0.0005A=0.5mA$.

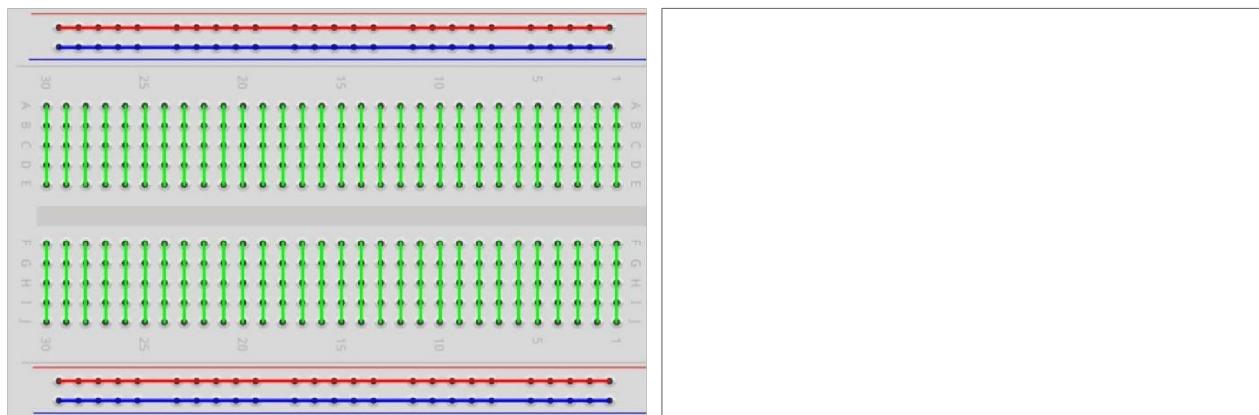


ADVERTENCIA: Nunca conecte los dos polos de una fuente de alimentación con algo de valor de resistencia bajo (es decir, un objeto de metal o un cable pelado), esto es un cortocircuito y genera una corriente alta que puede dañar la fuente de alimentación y los componentes electrónicos.

Nota: a diferencia de los LED y los diodos, las resistencias no tienen polos y no son polares (no importa en qué dirección las insertes en un circuito, funcionarán igual)

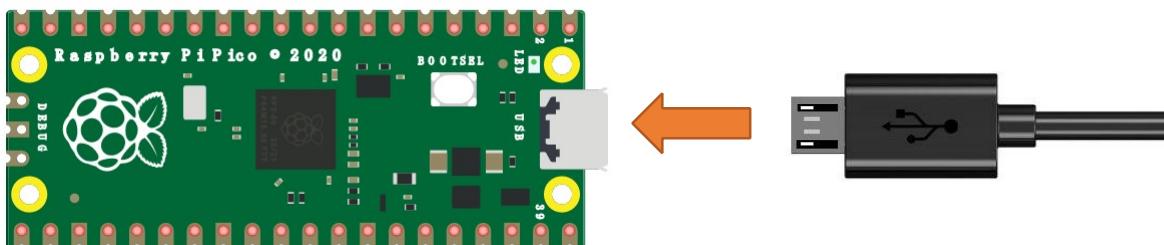
Tablero de circuitos

Aquí tenemos una pequeña placa de prueba como ejemplo de cómo se conectan eléctricamente las filas de orificios (enchufes). La imagen de la izquierda muestra la forma de conectar los pines. La imagen de la derecha muestra la estructura interna práctica.



Energía

En este tutorial, conectamos Raspberry Pi Pico y compter con un cable USB.

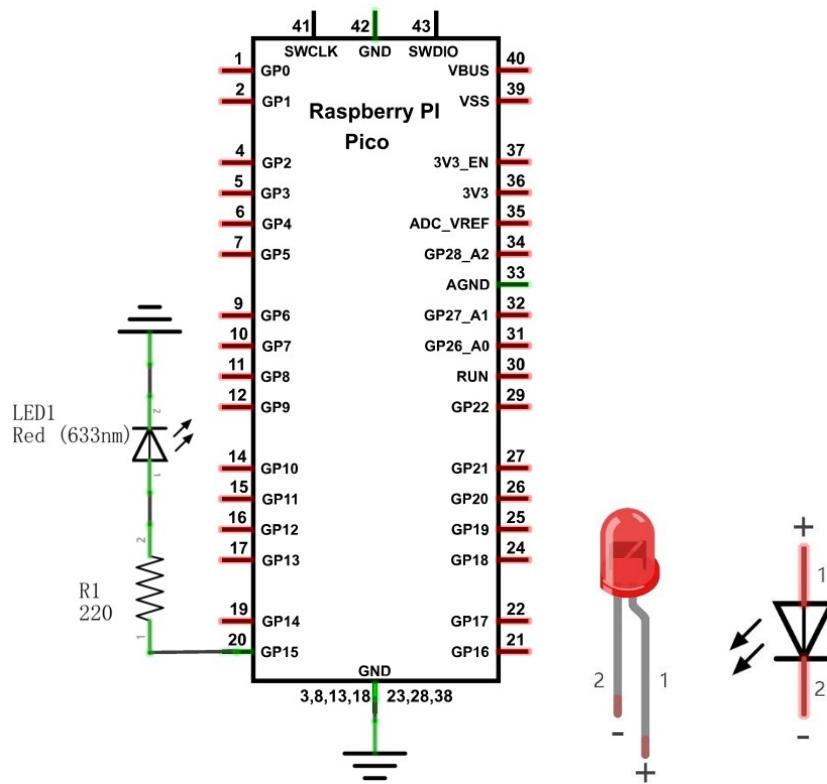


Circuito

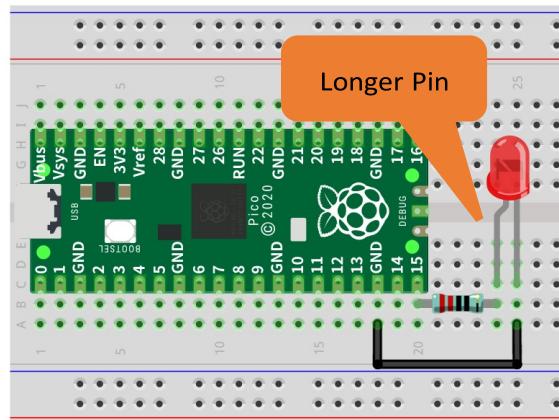
Primero, desconecte toda la energía del Raspberry Pi Pico. Luego construya el circuito de acuerdo con los diagramas de circuito y hardware. Después de que el circuito esté construido y verificado correctamente, conecte la PC a Raspberry Pi Pico.

PRECAUCIÓN: ¡Evite posibles cortocircuitos (especialmente conectando 3.3V y GND)! **ADVERTENCIA:** ¡Un cortocircuito puede causar una corriente alta en su circuito, crear un calor excesivo en los componentes y causar daños permanentes a su hardware!

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



Nota: Para ayudar a los usuarios a tener una mejor experiencia al realizar los proyectos, hemos realizado algunas modificaciones en el diagrama de simulación de Pico. Tenga en cuenta que existen ciertas diferencias entre el diagrama de simulación y la placa real para evitar malentendidos.

Código

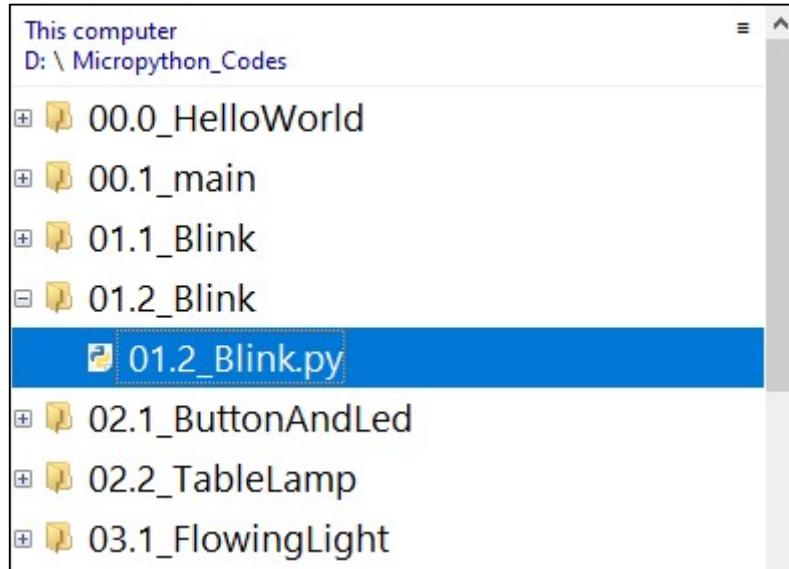
Los códigos utilizados en este tutorial se guardan en

“ [Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python_Codes](#) ”. Puede mover los códigos a cualquier ubicación. Por ejemplo, guardamos los códigos en Disk(D) con la ruta de “ [D:/Micropython_Codes](#) ”.

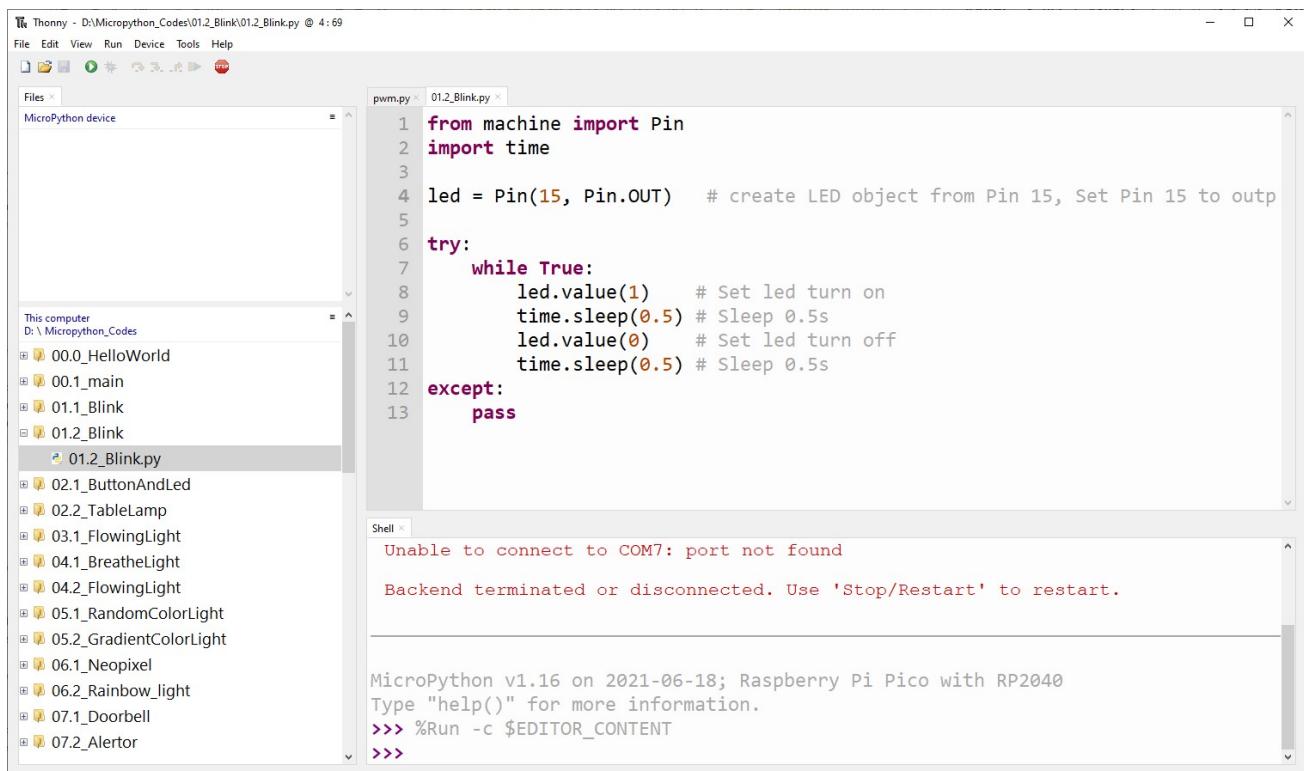
01.2_parpadeo

¿Cualquier duda? support@freenove.com

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes".



Expanda la carpeta "01.2_Blink" y haga doble clic en "01.2_Blink.py" para abrirla. Como se muestra en la siguiente ilustración.



Asegúrese de que Raspberry Pi Pico se haya conectado con la computadora. Haga clic en "Detener/Reiniciar backend" y luego espere a ver qué interfaz aparecerá.

```

from machine import Pin
import time

led = Pin(15, Pin.OUT) # create LED object from Pin 15, Set Pin 15 to output

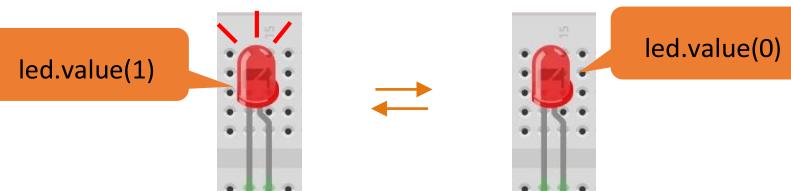
try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

```

Shell x
Unable to connect to COM7: port not found
Backend terminated or disconnected. Use 'Stop/Restart' to restart.

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
=>>> %Run -c \$EDITOR_CONTENT
=>>>

Haga clic en "Ejecutar secuencia de comandos actual" que se muestra en el cuadro de arriba, el código comienza a ejecutarse y el LED en el circuito comienza a parpadear. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



Nota :

Este es el código que se [ejecuta en línea](#). Si desconecta el cable USB y vuelve a encender Raspberry Pi Pico, el LED deja de parpadear y se mostrarán los siguientes mensajes en Thonny.

The screenshot shows the Thonny IDE interface. On the left, the file tree displays various Python scripts under 'D:\MicroPython_Codes'. The script '01.2_Blink.py' is selected and highlighted in blue. The main editor window shows the following code:

```

from machine import Pin
import time

led = Pin(15, Pin.OUT) # create LED object from Pin 15, Set Pin 15 to output

try:
    while True:
        led.value(1) # Set led turn on
        time.sleep(0.5) # Sleep 0.5s
        led.value(0) # Set led turn off
        time.sleep(0.5) # Sleep 0.5s
except:
    pass

```

Below the editor is a terminal window titled 'Shell' showing MicroPython v1.16 running on a Raspberry Pi Pico. The terminal output includes:

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
>>>
Connection lost (EOF)

Use Stop/Restart to reconnect.

```

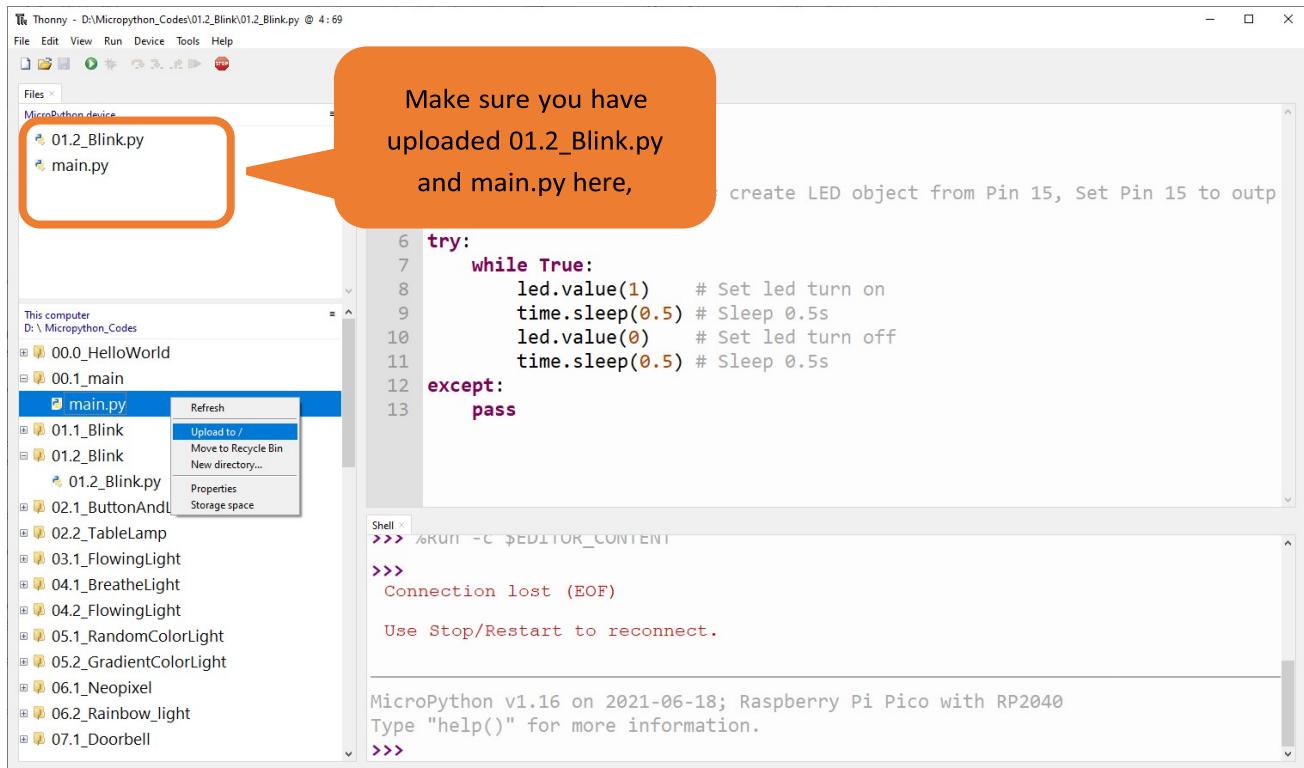
Subir código a Raspberry Pi Pico

Como se muestra en la siguiente ilustración, haga clic con el botón derecho en el archivo 01.2_Blink.py y seleccione "Cargar en /" para cargar el código en Raspberry Pi Pico.

This screenshot shows the Thonny IDE interface with the same code as the previous one. The file tree on the left shows '01.2_Blink.py' selected. A context menu is open over this file, with the 'Upload to /' option highlighted in blue. The menu also includes other options like 'Refresh', 'Move to Recycle Bin', 'New directory...', 'Properties', and 'Storage space'. The main editor and terminal windows are visible at the bottom.

Capítulo 1 LED (Importante)

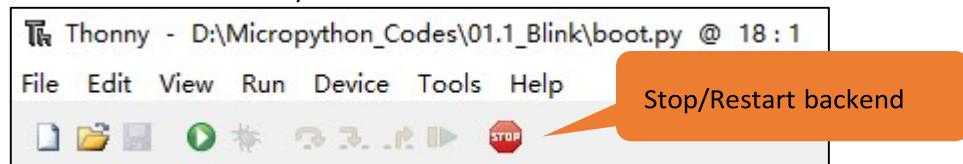
Upload main.py in the same way.



Desconecte el cable USB de Raspberry Pi Pico y vuelva a conectarlo, el LED en Pico parpadeará repetidamente.

Nota:

Los códigos aquí se ejecutan sin conexión. Si desea dejar de ejecutar sin conexión e ingresar a Shell, simplemente haga clic en "Detener" en Thonny.



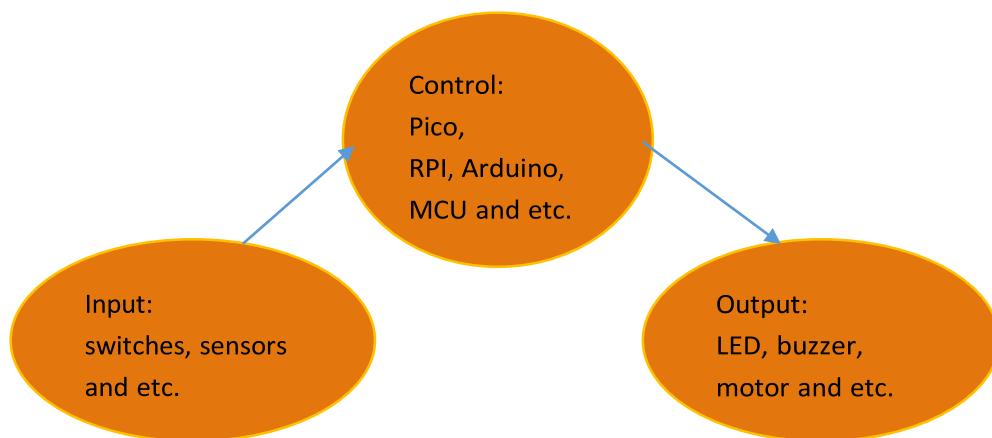
Si usted tiene alguna inquietud, contáctenos a través de: support@freenove.com

¿Cualquier duda?  support@freenove.com

Capítulo 2 Botón y LED

solo Raspberry Pi Pico y Raspberry Pi Pico W diferir por uno inalámbrica funcionan y son casi idéntico en otros aspectos _ En este tutorial, a excepción de la conexión inalámbrica función , otra Las piezas utilizan Raspberry Pi Pico mapa para demostración tutorial .

Por lo general , hay tres básicas partes en un dispositivo de control automático completo : ENTRADA , SALIDA y CONTROL. en último sección , el módulo LED era la parte de salida y Raspberry Pi Pico era la parte de control . en la práctica aplicaciones , nosotros no solamente hacer que los LED parpadeen , pero además hacer un dispositivo sentir el entorno medio ambiente , recibir instrucciones y luego toma la apropiada acción como los LED se encienden, hacen un zumbador encienda y así sucesivamente .



A continuación , nosotros hacer un proyecto simple : construir un sistema de control con botón , LED y Raspberry Pi Pico.

Entrada: Botón

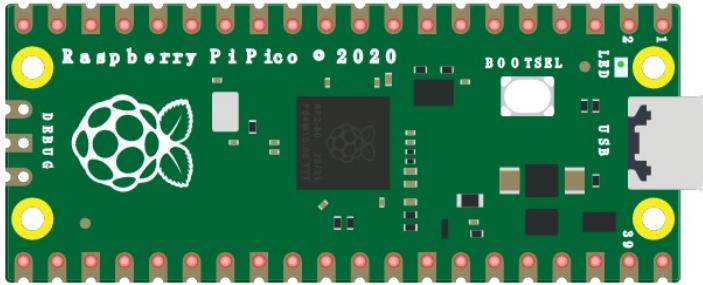
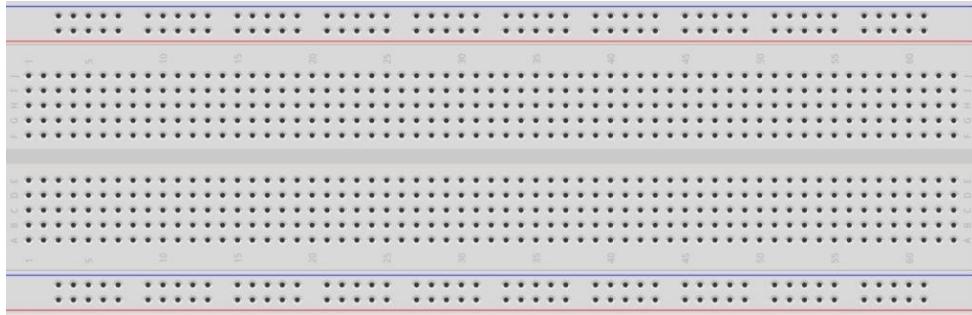
Control: Raspberry Pi Pico

Salida: LED

Proyecto 2.1 Botón y LED

En el proyecto , nosotros controlará el estado del LED a través de un empuje Interruptor de botón . Cuando el botón es presionado , nuestro LED _ encienda , y cuando eso es liberado , el LED apagar . Esto describe un interruptor momentáneo .

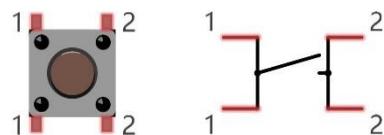
Lista de componentes

Frambuesa Pi Pico x1		Cable USBx1		
Protopboard x1				
Puente	LED x1	Resistor 220Ω x1	Resistor 10kΩ x2	Empujar botón x1

Componente conocimiento

Empujar botón

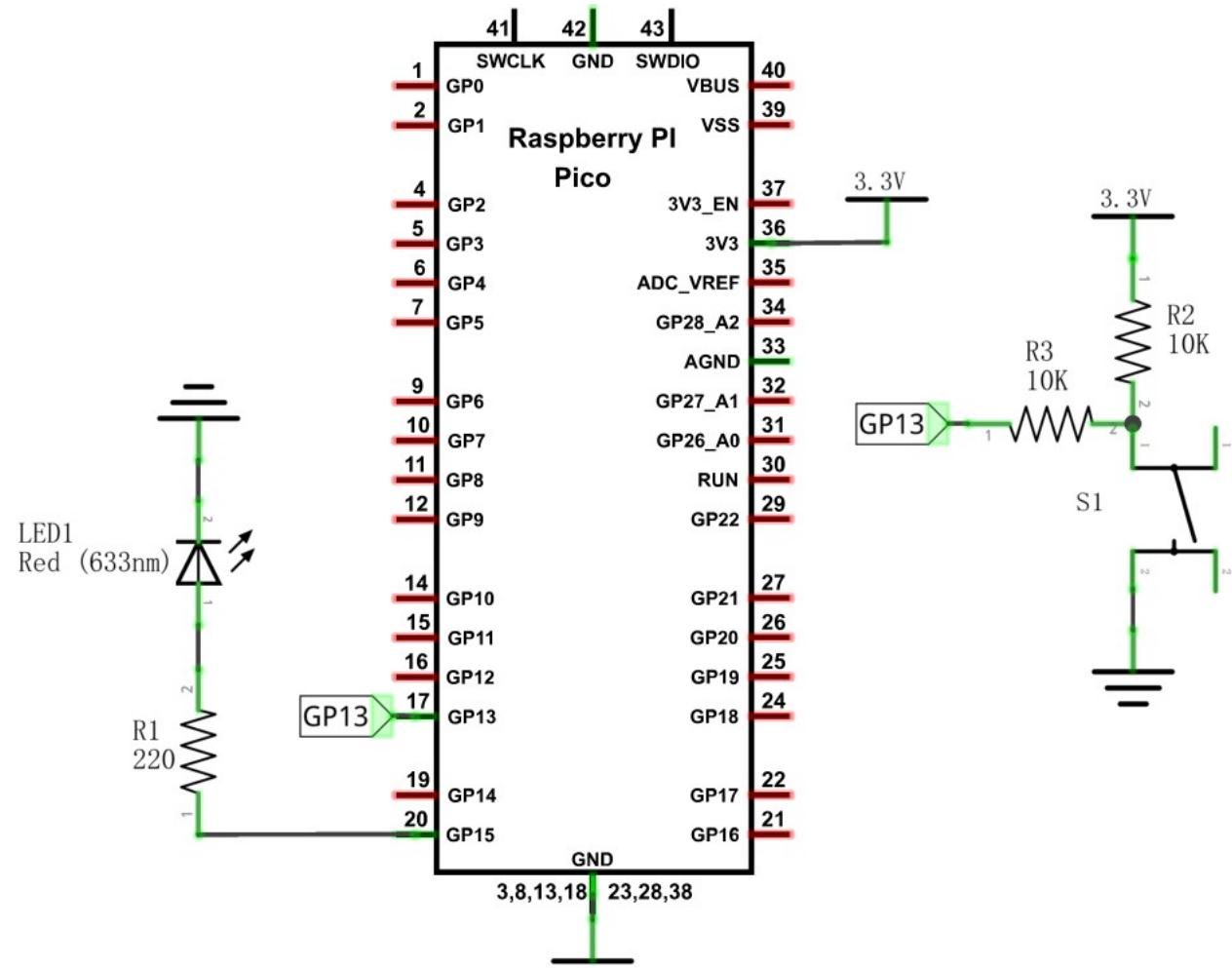
Este tipo de empuje El interruptor de botón tiene 4 pines (interruptor de 2 polos). Dos patas a la izquierda están conectados , y ambos izquierda y derecha los lados son iguales segúin la ilustración :



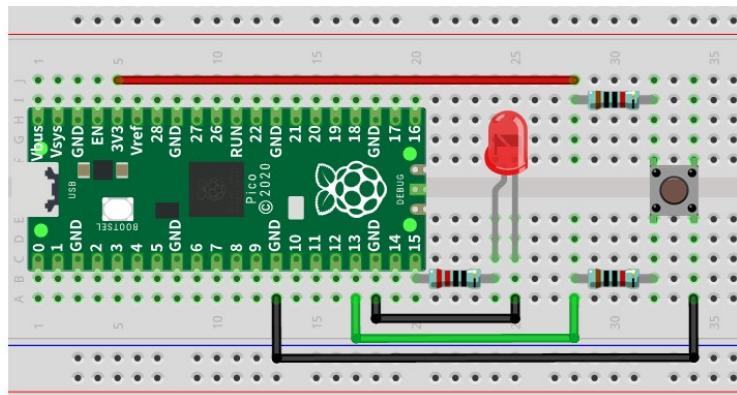
Cuando el botón en el interruptor está presionado , el circuito es completado (su proyecto es encendido).

Circuito

Schematic diagram



Hardware connection. If you need any support, please free to contact us via: support@freenove.com



Note: To help users have a better experience when doing the projects, we have made some modifications to Pico's simulation diagram. Please note that there are certain differences between the simulation diagram and the actual board to avoid misunderstanding.

Código

Este proyecto es diseñado a aprender para controlar un LED con una pulsación interruptor de botón Primero , nosotros necesitar a lea el estado del interruptor y luego decida si el LED está transformado en o no establecido en eso _

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico / Python_Codes** " al disco (D) por adelantado con la ruta de " **D:/ Micropython_Codes** " .

Abra "Thonny", haga clic en " Esta computadora " → " D:" → " Micropython_Codes " → " 02.1_ButtonAndLed " y doble haga clic en " 02.1_ButtonAndLed.py ".

02.1_ButtonAndLed

```

from machine import Pin
import time

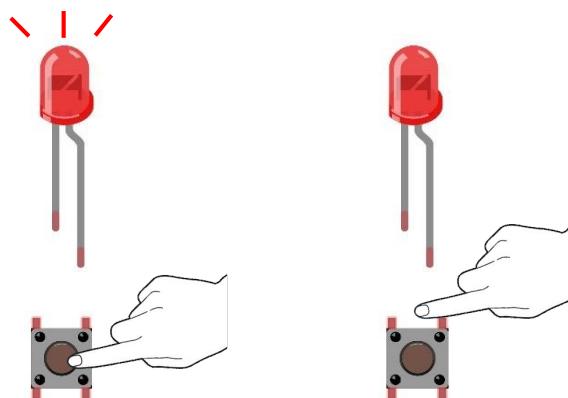
led = Pin(15, Pin.OUT)
button = Pin(13, Pin.IN, Pin.PULL_UP) #Create button object from Pin13 ,

try:
    while True:
        if not button.value():
            led.value(1) #Set led turn on
        else:
            led.value(0) #Set led turn off
except:
    pass

```

The Thonny interface includes a toolbar with icons for file operations, a status bar at the bottom, and a shell window at the bottom right showing MicroPython version information.

Haga clic en "Ejecutar secuencia de comandos actual" que se muestra en el cuadro de arriba ilustración , presione el botón interruptor de botón , el LED se enciende ; suelte el interruptor, el LED se apaga . Prena Ctrl + C o haga clic en "Detener/ Reiniciar back- end " a salida programa _



Lo siguiente es el programa código :

```

1 de importación de máquinas
2 Alfiler importar tiempo
3 led = Pin ( 15, Pin. SALIDA )           botón = Pin(13, Pin. IN, Pin. PULL_UP )
4 Botón #Crear objeto de Pin13, Pin13 a la entrada
5 prueba :     mientras que es
6 cierto :       si no botón.
7 valor ( ):
8         dirigió. valor ( 1 )           #Establecer
9 giro led en     más :
10        dirigió. valor ( 0 )           #Establecer
11 apagado del led excepto :      pasar
12
13
14

```

en este proyecto , usamos el módulo Pin de la máquina, así que antes inicializando el Pin , necesitar a importar este módulo primero .

1	de importación de máquinas Alfiler
---	------------------------------------

en el circuito Conexión , LED y botón están conectados con GP15 y GP13 respectivamente , defina el LED y el botón como 0 y 3 respectivamente .

4	led = Pin (15, Pin. SALIDA)
5	botón = Pin(13, Pin. IN, Pin. PULL_UP) Botón #Crear objeto desde Pin13, establezca Pin13 en la entrada

Leer el estado del pin del botón con función de valor () . Presione el interruptor de botón , la función devoluciones bajo nivel y el resultado de " si " es verdadero, y luego el LED se encenderá ; De lo contrario , el LED es APAGADO .

```

9      si no botón. valor ( ) :
10     dirigió. valor ( 1 )           #Establecer
11   giro led en     más :
12     dirigió. valor ( 0 )         #Establecer apagado del led

```

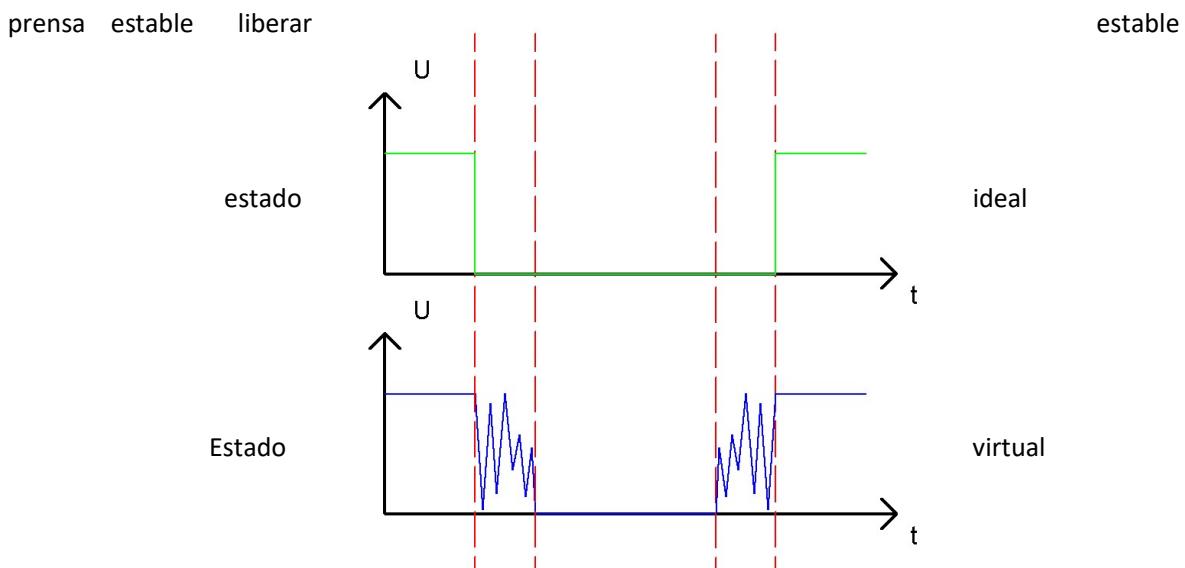
Lámpara de sobremesa Proyecto 2.2 MINI

Nosotros voluntad también usa un Empuje Interruptor de botón , LED y Raspberry Pi Pico para hacer una lámpara de mesa MINI pero este voluntad función de manera diferente : presione el botón , el LED encienda , y presionando el botón de nuevo , el LED se apaga . La acción del interruptor ON ya no es momentáneo (como una puerta campana) pero permanece ENCENDIDO sin necesitando a continuamente prensa en el interruptor de botón .

Primero , deja a nosotros aprender alguna cosa sobre el empuje interruptor de botón

Rebote para empujar Botón

Cuando un momento Empujar El interruptor de botón es presionado , es voluntad no cambio de una estado a otro estado inmediatamente _ Adeudado a diminuto mecánico vibraciones , hay será un breve período de continua sacudiendo antes de eso completamente alcanza otro estado también rápido para los humanos a detectar pero no para microcontroladores de computadora . lo mismo es cierto cuando el empuje interruptor de botón es liberado _ Este no deseado fenómeno es conocido como “ rebote ” .



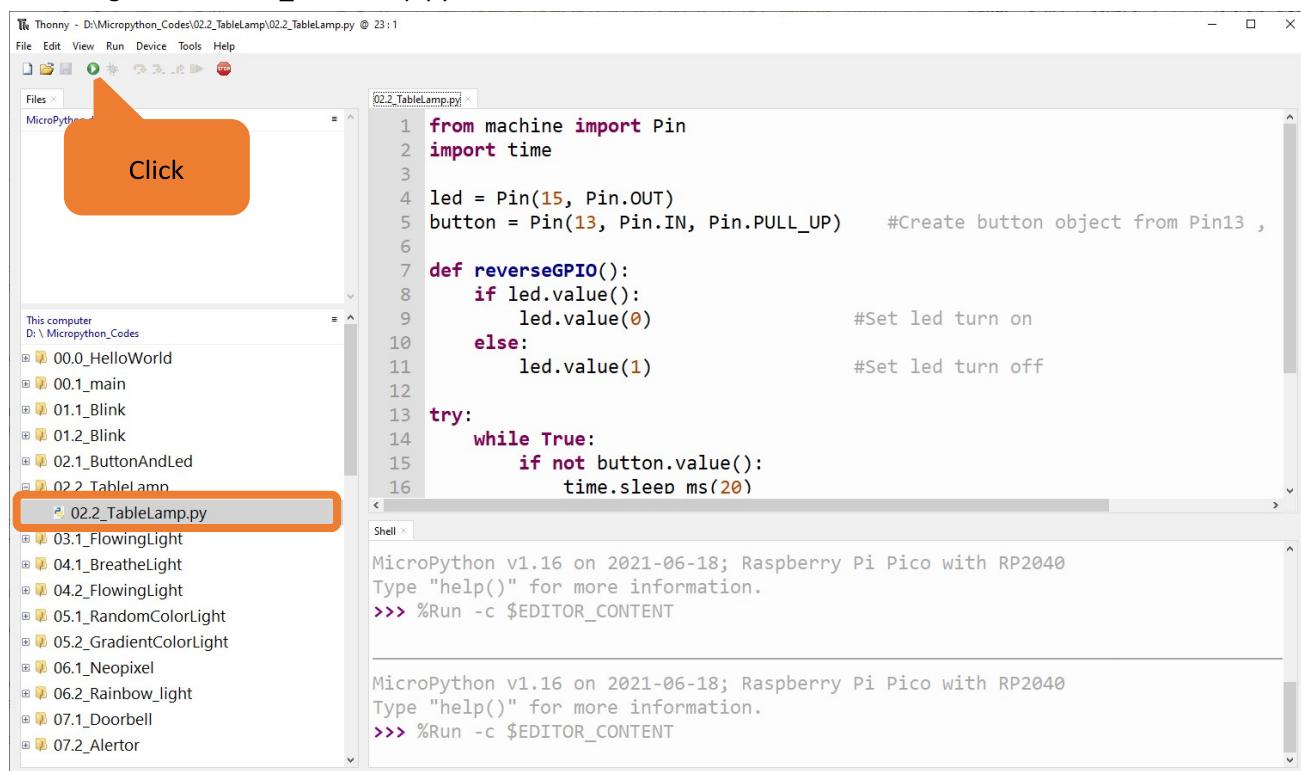
Por lo tanto , si podemos directamente _ detectar el estado del Push Interruptor de botón , hay múltiples presiones y liberaciones acciones en un ciclo de prensado . Este sacudiendo voluntad engañar a la alta velocidad funcionamiento del microcontrolador para causar muchas decisiones falsas . Por lo tanto , nosotros necesitar a eliminar el impacto de los golpes . Nuestro solución : a juzgar el estado del botón varias veces Solamente cuando el botón estado es estable (consistente) durante un período de tiempo, ¿ puede indicar que el botón es en realidad en el estado ON (siendo presionado) . Este proyecto necesita lo mismo componentes y circuitos como nosotros utilizado en el anterior sección _

Código

02.2_Lampara de mesa

Mueva la carpeta del programa " Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico / Python_Codes " al disco (D) por adelantado con la ruta de " D:/ Micropython_Codes ".

Abra " Thonny ", haga clic en " Esta computadora " → " D:" → " Micropython_Codes " → " 02.2_TableLamp " y doble haga clic en " 02.2_TableLamp.py ".



```

from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(13, Pin.IN, Pin.PULL_UP)      #Create button object from Pin13 ,

def reverseGPIO():
    if led.value():
        led.value(0)                      #Set led turn on
    else:
        led.value(1)                      #Set led turn off

try:
    while True:
        if not button.value():
            time.sleep_ms(20)

```

Shell x

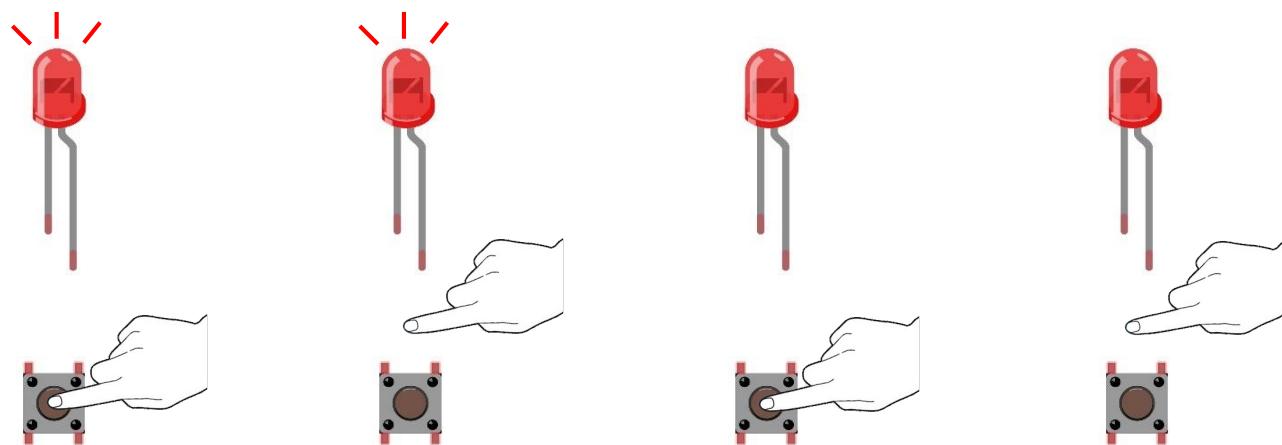
```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

```

Haga clic en "Ejecutar secuencia de comandos actual " que se muestra en el cuadro de arriba ilustración , presione el botón interruptor de botón , LED gira EN ; prensa eso de nuevo , el LED se apaga . Prena Ctrl + C o haga clic en "Detener/ Reiniciar back- end " a eixt archivo.



Si tú tener ningún preocupaciones , por favor contacto a nosotros vía : support@freenove.com

Lo siguiente es el programa código :

```

1 de importación de
2 máquinas Alfiler importar
3 tiempo
4 led = Pin ( 15, Pin.SALIDA )           botón = Pin(13, Pin.IN, Pin.PULL_UP )
5 Botón #Crear objeto desde Pin13, establezca Pin13 en la entrada
6
7 definitivamente
8 e GPIO inverso
9 (): si
10 led.valor () :
11     dirigió. valor ( 0 )
12 #Establecer giro led en más :
13     dirigió. valor ( 1 )           #Establecer apagado del led
14 prueba : mientras que es
15 cierto : si no botón.
16 valor () :
17 tiempo.dormir_ms (20)
18 si no botón.valor () :
19     GPIO inverso ( )
20 tiempo no botón.valor () :
21     time.sleep
22     _ms (20) excepto : pasar

```

Cuando el botón es detectado para ser presionado , demora 20ms para evitar el efecto de rebote , y luego verifique si el botón ha sido presionado otra vez Si es así, el condicional declaración se ejecutará , de lo contrario eso voluntad no ser ejecutado .

```

14 tiempo cierto :
15     si no botón. valor
16 dieciséis (): time.sleep_ms
17     (20) si no
18     botón.valor () :
19         GPIO inverso ( )
20     mientras no botón.valor () :
21         tiempo.sueño_ms (20)

```

Personaliza una función y un nombre eso reverseGPIO (), que invierte el nivel de salida del LED.

```

7 definitivamente GPIO
8 inverso () : si
9 led.valor () :
10     dirigió. valor ( 0 )           #Establecer
11 giro led en más :
12     dirigió. valor ( 1 )           #Establecer apagado del led

```

Capítulo 3 Barra LED

solo Raspberry Pi Pico y Raspberry Pi Pico W diferir por uno inalámbrica funcionan y son casi idéntico en otros aspectos _ En este tutorial, a excepción de la conexión inalámbrica función , otra las piezas utilizan Raspberry Pi Pico mapa para demostración tutorial .

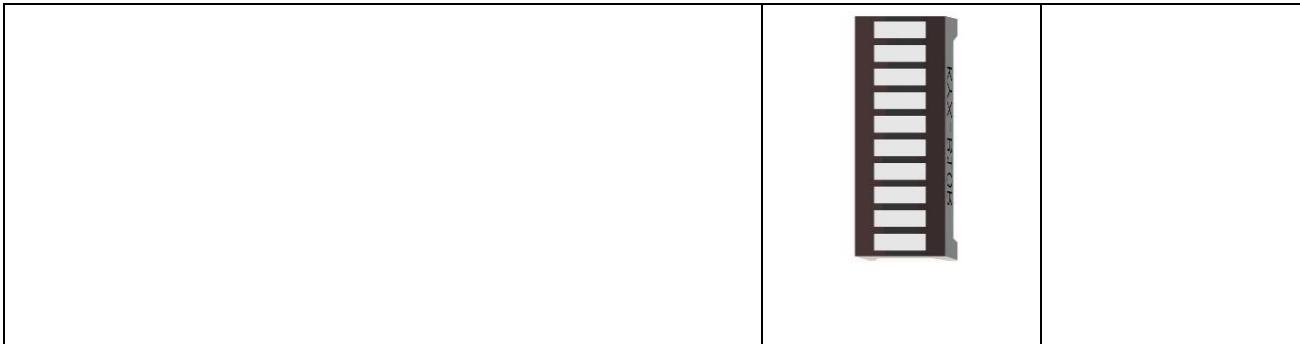
Nosotros tener aprendido cómo para controlar el parpadeo de un LED , a continuación nosotros voluntad aprender cómo para controlar varios LED.

Proyecto 3.1 Luz que fluye

en este proyecto , utilizamos una serie de LED para hacer una luz que fluye .

Lista de componentes

Frambuesa Pi Pico x1	Cable USBx1	
		
Protoboard x1		
Puente	Gráfico de barras LED x1	Resistencia 220Ω x10
		

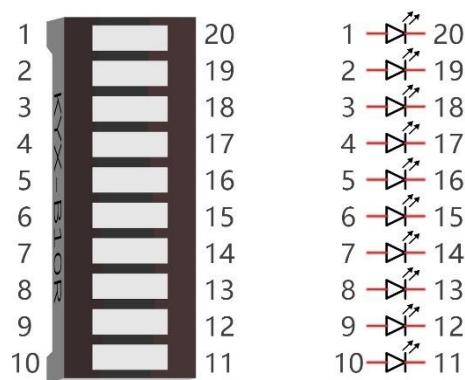


Componente conocimiento

Dejar a nosotros aprender sobre lo basico caracteristicas de estos componentes para usar y entender a ellos mejor

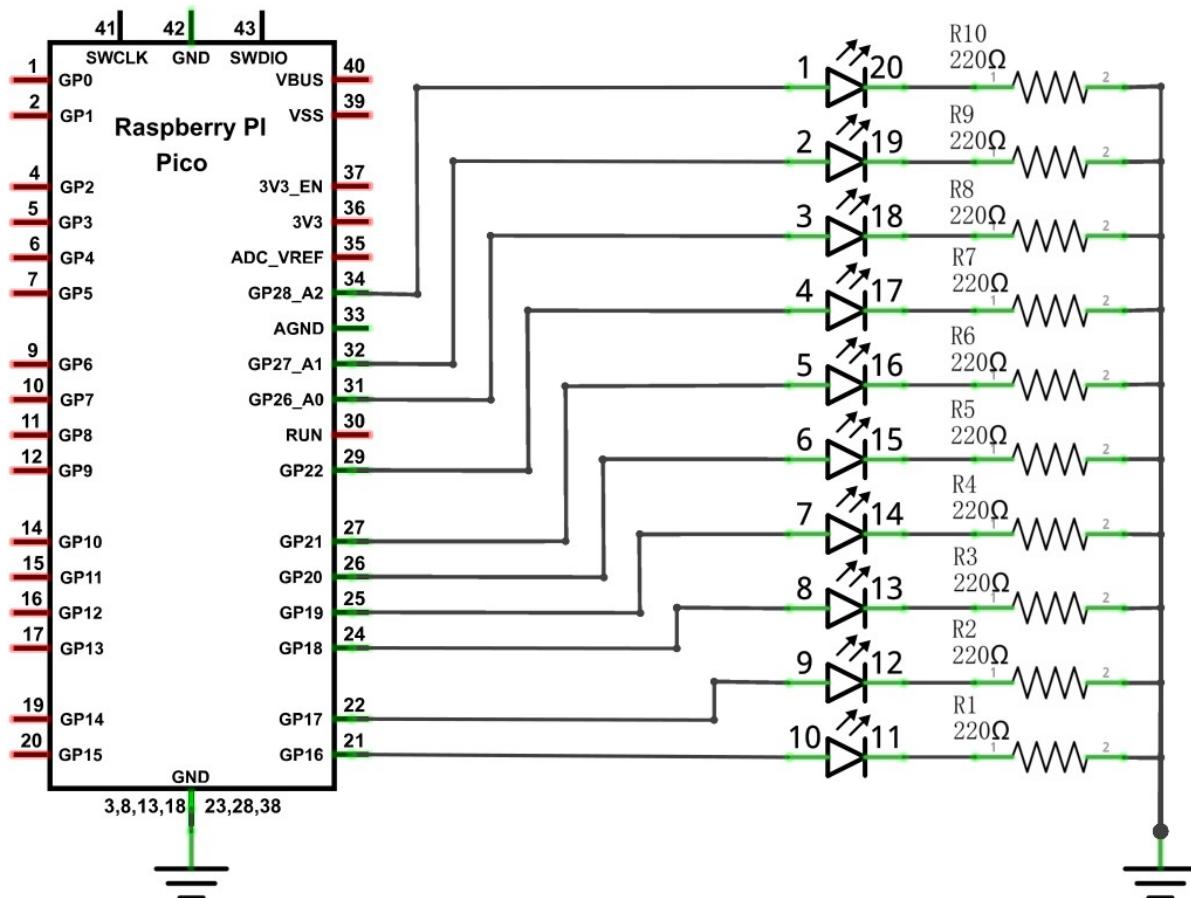
— barra de leds

gráfico de barras tiene 10 LED integrados dentro un componente compacto . los dos las filas de pines en su parte inferior están emparejadas a identificar cada LED como el único LED utilizado antes _

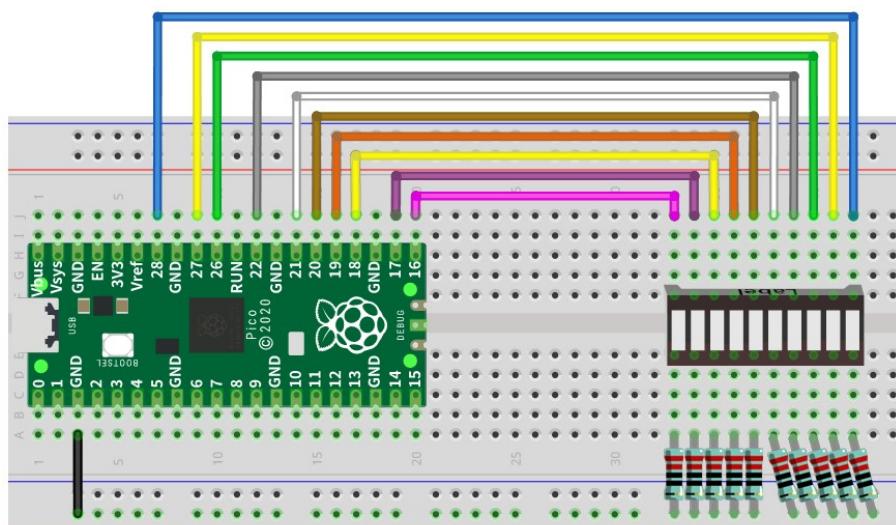


Circuito

Esquemático diagrama



Conexión de hardware . Si tú necesitar ningún apoyo , por favor siéntete libre de contacto a nosotros vía :
support@freenove.com



Nota: Para ayuda usuarios tener un mejor experiencia cuando haciendo los proyectos , nosotros tener hecha alguno modificaciones a de pico simulación diagrama _ Tenga en cuenta que hay ciertos _ diferencias entre la simulación diagrama y el tablero real a evitar malentendido _

Si Barra LED no trabajar , tratar de girar Barra LED para 180°. la etiqueta es al azar

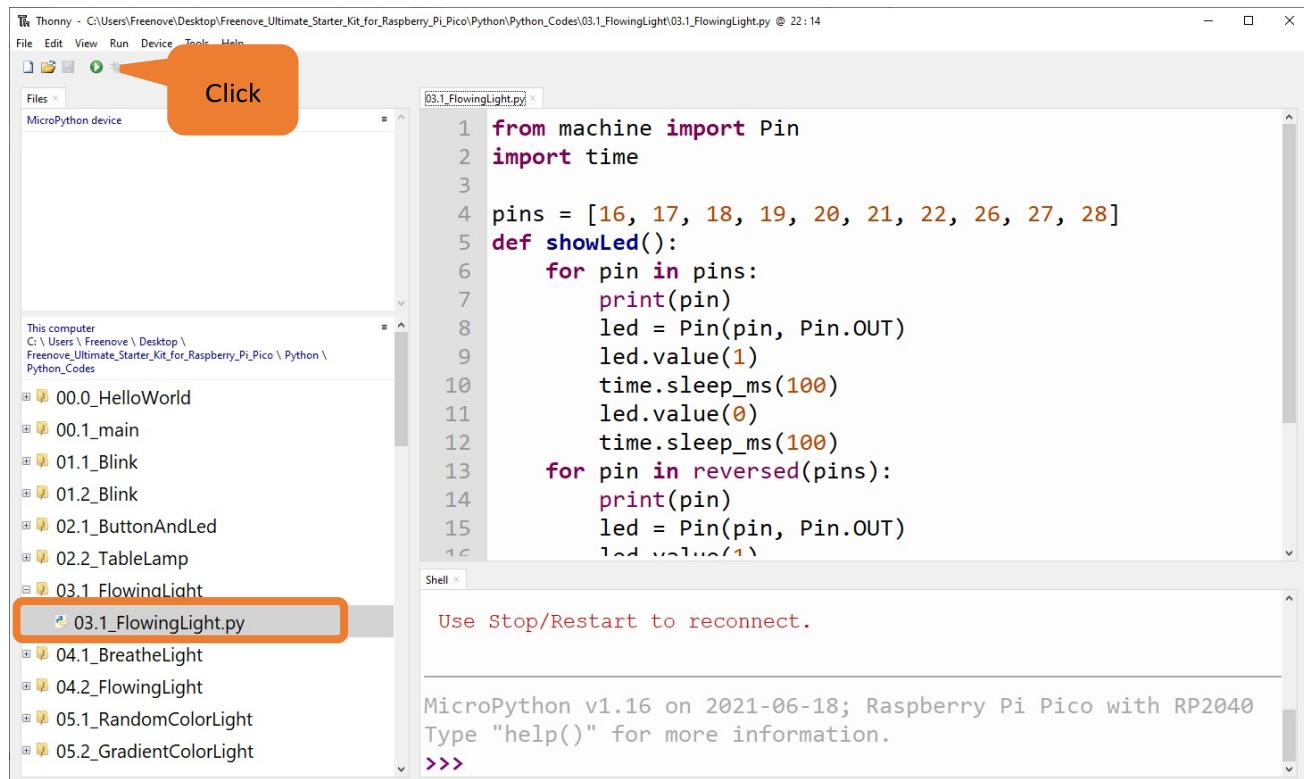
Código

Este proyecto es diseñado a hacer un fluir agua lampara _ cuales son estos acciones : Primero encienda el LED #1, luego giro está APAGADO. Despues encienda el LED #2, y luego giro APAGADO... y repetir lo mismo a los 10 LED hasta que se encienda el ultimo LED . se apaga . Este proceso es repetido a lograr los “ movimientos ” de fluir agua

03.1_Luz que fluye

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico / Python_Codes** " al disco (D) por adelantado con la ruta de " **D:/ Micropython_Codes** ".

Abra " Thonny ", haga clic en " Esta computadora " → " D:" → " Micropython_Codes " → " 03.1_FlowingLight " y doble haga clic en " 03.1_FlowingLight.py ".



```

from machine import Pin
import time

pins = [16, 17, 18, 19, 20, 21, 22, 26, 27, 28]
def showLed():
    for pin in pins:
        print(pin)
        led = Pin(pin, Pin.OUT)
        led.value(1)
        time.sleep_ms(100)
        led.value(0)
        time.sleep_ms(100)
    for pin in reversed(pins):
        print(pin)
        led = Pin(pin, Pin.OUT)
        led.value(1)

```

Use Stop/Restart to reconnect.

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar secuencia de comandos actual " que se muestra en el cuadro de arriba , gráfico de barras LED se iluminará desde izquierda a a la derecha y luego hacia atrás Correcto a izquierda _ Prena Ctrl + C o haga clic en "Detener/ Reiniciar back- end " a salir del programa .



Si tú tener ningún preocupaciones , por favor contacto a nosotros vía : support@freenove.com

lo siguiente es el programa código :

Ningún preocupaciones ?  support@freenove.com

```

1 de importación de máquinas
2 Importación de pines
3 tiempo
4 pines = [16, 17, 18, 19, 20, 21, 22, 26,
5 27, 28] def showLed (): para pin en pines
6 : led = Pin(pin, Pin.OUT) led.value (1)
7 time.sleep_ms (100) led.value (0)
8 time.sleep_ms (100) for pin in reverse (
9 pins ) : led = Pin(pin, Pin.OUT) led.valor
10 (1) time.sleep_ms (100) led.value (0)
11 time.sleep_ms (100)
12 while
13 True :
14 showLed (
15 )
16 dieciséis
17
18
19
20

```

Use una matriz para definir 10 puertos GPIO conectado al gráfico de barras LED para facilitar operación _

4	pines = [16, 17, 18, 19, 20, 21, 22, 26, 27, 28]
---	--

Usar dos para bucles a giro en LED por separado de izquierda a la derecha y luego hacia atrás Correcto a izquierda

```

5 definitivamente showLed (
6 ): para pin en pines : led
7 = Pin(pin, Pin.OUT)
8 led.value (1) time.sleep_ms
9 (100) led.value (0)
10 time.sleep_ms (100) for pin
11 in reverse ( pins ) : led =
12 Pin(pin, Pin.OUT)
13 led.valor (1) time.sleep_ms
14 (100) led.value (0)
15 time.sleep_ms (100)
16 dieciséis
17

```

Referencia

por	
-----	--

Para bucle es usó a ejecutar un programa infinitamente e iterar en el orden de los elementos (una lista o una cadena) en la secuencia . Comúnmente usado :

“para alfiler en alfileres ”

Patas es una lista de elementos que se iteran sobre por un bucle for y asignado para fijar cada vez.

“for i in range (start , end , num : 1)” start : initial valor , de que el bucle for empieza contando _ La inicial por defecto valor es 0. Por ejemplo , el rango (5) es igual a a rango (0, 5).

fin : el valor con cual la función termina _ Para bucle cuenta hasta eso llega a esto valor , pero este valor no es incluido en el conteo . número : número es automáticamente adicional cada vez a los datos. El valor predeterminado es 1. función de rango () devuelve una secuencia número cual es asignado a I por bucle for .

Capítulo 4 Analógico y PWM

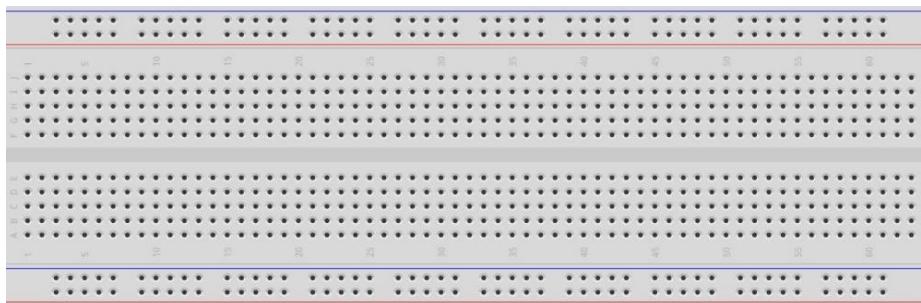
En anterior estudiar , nosotros tener conocido que una el botón tiene dos estados : presionado y liberado , y el LED tiene un estado de encendido / apagado , luego cómo a entrar en un medio estado ? Cómo para dar salida a un intermedio estado a dejar LED "semi brillante "? Ese es qué fueron yendo a aprender _

Primero , vamos aprender cómo para controlar el brillo de un LED.

Proyecto 4.1 LED de respiración

Respirando luz, que es , LED es transformado de apagado a en gradualmente , y gradualmente de en a apagado, al igual que " respiración ". Entonces, ¿cómo Cómo controlar el brillo de un LED ? Nosotros utilizará PWM para lograr este objetivo

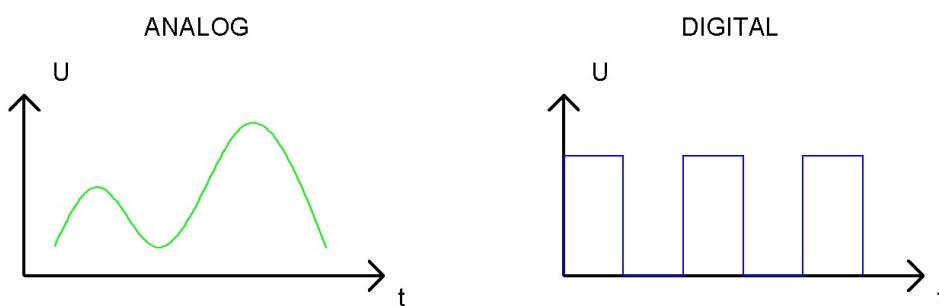
Lista de componentes

Frambuesa Pi Pico x1	Cable USBx1	
		
Protoboard x1		
		
LED x1	Resistencia 220Ω x1	Puente
		

Relacionado conocimiento

analógico y digital

Un Cosa análoga Señal es un continuo señal tanto en tiempo como en valor . Por el contrario , una Señal Digital o tiempo discreto señal es una serie de tiempo que consta de una secuencia de cantidades . La mayoría Las señales en la vida son analógicas. señales _ Un ejemplo familiar de un Cosa análoga Señal seria como seria la temperatura durante todo el día es continuamente cambiando y podría no de repente cambio instantáneamente de 0°C a 10°C. Sin embargo , Digital Signals puede instantáneamente cambio de valor . Este cambio es expresado en números como 1 y 0 (la base de binario código). Sus Las diferencias se pueden ver más fácilmente . cuando comparado cuando graficado como se muestra a continuación .



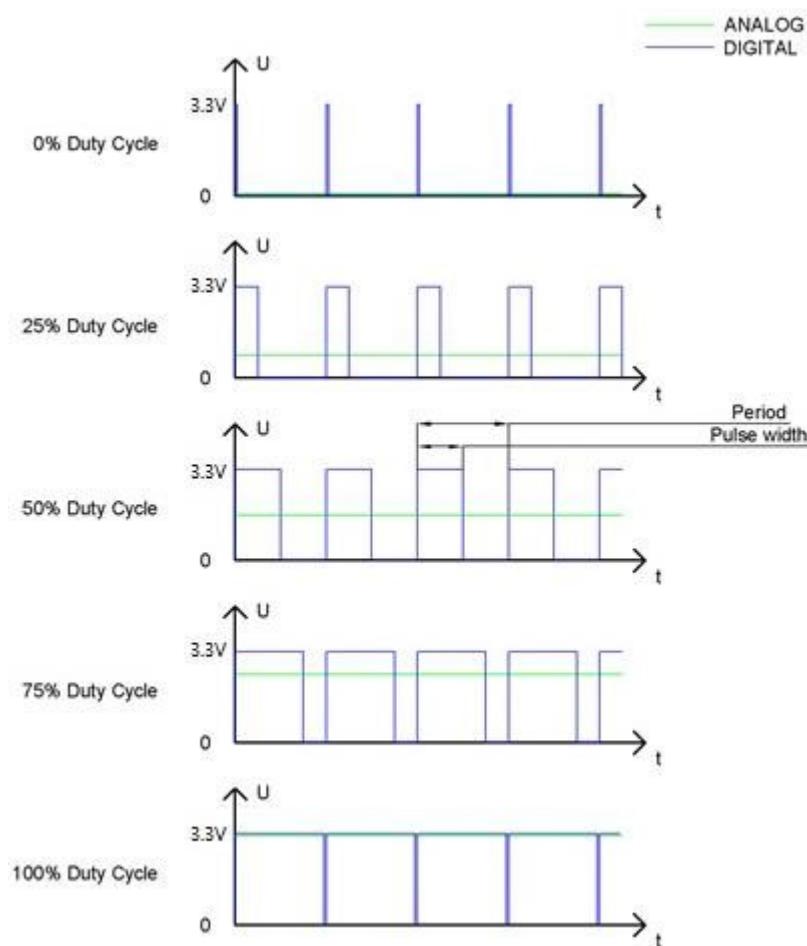
en la práctica aplicación , nosotros a menudo usa binario como la señal digital , que es una serie de 0 y 1. Desde un binario señal solo tiene dos valores (0 o 1) , tiene gran estabilidad y confiabilidad . Por último , ambos Las señales analógicas y digitales se pueden convertir. en el otro .

PWM

PWM, ancho de pulso La modulación , es muy eficaz método para usar señales digitales para controlar analógico circuitos _ Común procesadores no poder salida analógica directa señales _ tecnología PWM hace eso muy conveniente a lograr este conversión (traducción de digital a cosa análoga señales)

tecnología PWM utiliza pines digitales a enviar cierto frecuencias de cuadrado olas , que es , la salida de alta niveles y bajos niveles , que alternativamente durar un tiempo . El tiempo total para cada conjunto de alta niveles y bajos niveles es en general fijo , que es llamado período (Nota: el recíproco del período es frecuencia). El tiempo de alta las salidas de nivel son generalmente llamado " ancho de pulso " , y el deber ciclo es el porcentaje de la relación entre la duración del pulso o el ancho del pulso (PW) y el período total (T) de la forma de onda .

Cuanto más larga sea la salida de alta niveles último , cuanto más largo sea el deber ciclo y cuanto mayor sea el correspondiente voltaje en el analogico señal será _ Las siguientes figuras muestran cómo el análogo señal voltajes variar entre 0V-5V (alto nivel es 5V) correspondiente al ancho de pulso 0%-100%:



Cuanto más largo sea el deber de PWM ciclo es , cuanto mayor sea la potencia de salida será _ Ahora que nosotros comprenden este relación , podemos utilizar PWM para controlar el brillo de un LED o la velocidad del motor de CC y así sucesivamente . Eso es evidente de lo anterior que PWM es no analógico real , y el efectivo valor del voltaje es equivalente a la correspondiente analógico _ por lo tanto, podemos controlar la potencia de salida del LED y otros módulos de salida para lograr diferente efectos _

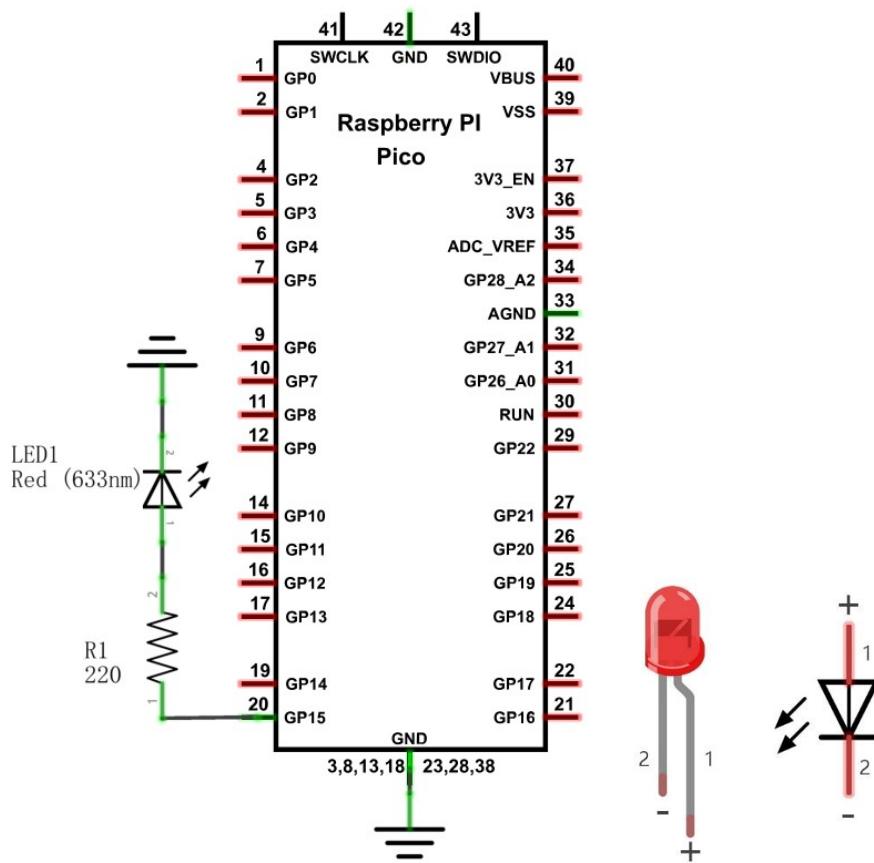
Raspberry Pi Pico y PWM

Raspberry Pi Pico tiene 16 canales PWM , cada uno de los cuales puede controlar la frecuencia y el trabajo ciclo independientemente , con el reloj frecuencia rangos de 7Hz a 125MHz . Cada pin en Raspberry Pi Pico se puede configurar como salida PWM.

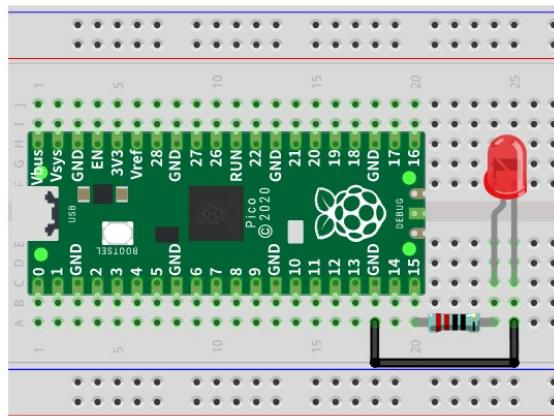
Circuito

Este circuito es el mismo que el del proyecto parpadeo _

esquemático _ diagrama



Conexión de hardware . Si tú necesitar ningún apoyo , por favor siéntete libre de contacto a nosotros vía :
support@freenove.com



Nota: Para ayuda usuarios tener un mejor experiencia cuando haciendo los proyectos , nosotros tener hecha alguno modificaciones a de pico simulación diagrama _ Tenga en cuenta que hay ciertos _ diferencias entre la simulación diagrama y el tablero real a evitar malentendido _

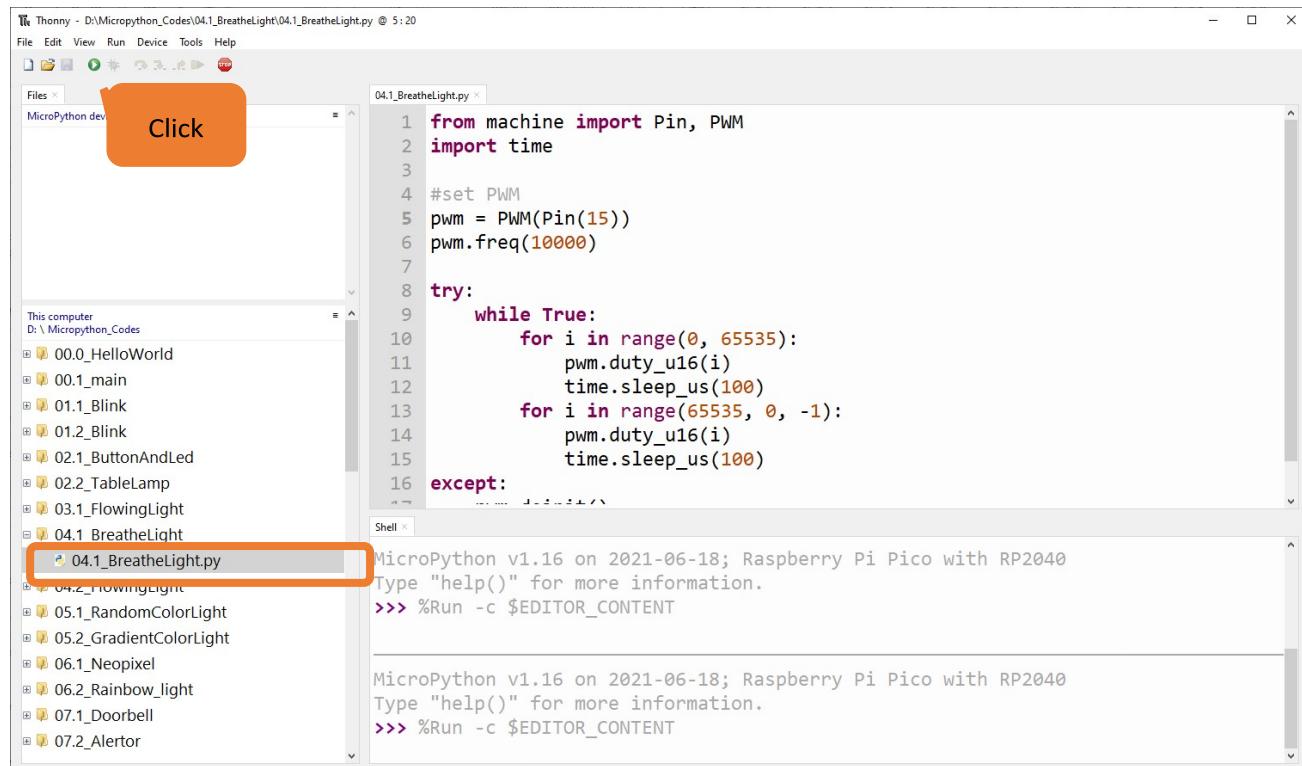
Código

Este proyecto es diseñado a hacer salida PWM GP15 con ancho de pulso creciente de 0% a 100%, y luego reduciendo del 100% al 0% gradualmente .

Ningún preocupaciones ? support@freenove.com

Abra “Thonny”, haga clic en “Esta computadora” → “D:” → “Micropython_Codes” → “04.1_BreatheLight” y doble haga clic en "04.1_BreatheLight.py".

04.1_BreatheLight



```

from machine import Pin, PWM
import time

#set PWM
pwm = PWM(Pin(15))
pwm.freq(10000)

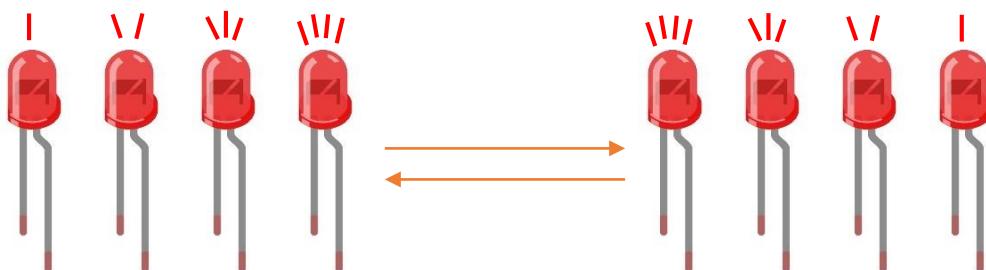
try:
    while True:
        for i in range(0, 65535):
            pwm.duty_u16(i)
            time.sleep_us(100)
        for i in range(65535, 0, -1):
            pwm.duty_u16(i)
            time.sleep_us(100)
except:
    ...

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
=>>> %Run -c \$EDITOR_CONTENT

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
=>>> %Run -c \$EDITOR_CONTENT

Haga clic en "Ejecutar secuencia de comandos actual " y verá ver ese led es transformado de ENCENDIDO a APAGADO y luego de APAGADO a ENCENDIDO gradualmente me gusta respiración _ Prena Ctrl + C o haga clic en "Detener/ Reiniciar back- end " a salida programa _



lo siguiente es el programa código :

```

1     de importación de
2     máquinas Pin,
3     importación PWM
4     tiempo
5     #establecer
6     PWM pwm =
7     PWM ( Pin (
8     15))
9     pwm.freq
10    (10000)
11    intente : while True :
12    for i in range ( 0,
13    65535):
14        pwm.duty_u16(i)
15        time.sleep_us (100) para i en
16        rango (65535, 0, -1):
17            pwm.duty_u16(i)
18            time.sleep_us (100) excepto :
19                pwm.deinit ()

```

Cree un objeto PWM y configure GP15 como pin de salida PWM. Llamar freq () para configurar la frecuencia de salida PWM de GP15 a 10000Hz.

5	pwm = PWM (Pin (15)) pwm.freq
6	(10000)

El rango del deber ciclo es 0-65535, por lo que usamos el primer ciclo for para controlar PWM para cambiar el deber ciclo valor , haciendo salida PWM 0% -100%; Usa el segundo bucle for a hacer salida PWM 100%-0%.

10	para i en el rango (0, 65535):
11	pwm.duty_u16(i)
12	time.sleep_us (100) para i en rango
13	(65535, 0, -1):
14	pwm.deber_u16(i) tiempo.dormir_us
15	(100)

Cada vez que PWM es utilizado , el temporizador de hardware se encenderá para __ cooperar eso _ Por lo tanto , después de cada uso de PWM, deinit () necesita ser llamado _ a apagó el temporizador . De lo contrario , el PWM puede fallar a trabajar la próxima vez

17	pwm.deinit ()
----	---------------

Referencia

Clase PWM (clavija)	
----------------------	--

B

el archivo pitón . 'WM, por favor agregue la declaración " **de máquina importar PWM** " en la parte superior de

pin : se admiten pines PWM ,

PWM.freq (freq_val) :

GP (0 ~ 22) 、 GP (25) 、 GP (26 ~ 28).

PWM.deber_u16(deber_val) ión es usó para configurar la frecuencia PWM y los retornos nada ; cuando allá no es

) :

0 y devuelve la frecuencia PWM . la función es usó para establecer el servicio PWM es
aún se ha establecido, es clo , entre que , valor_deber rangos de

devuelve 0. ámetro , la función devoluciones a deber establecido actualmente ciclo _ Si deber

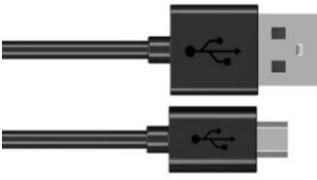
PWM.deinit () : Apague PWM.

ciclo no tiene

Proyecto 4.2 Flujo de luz de meteorito

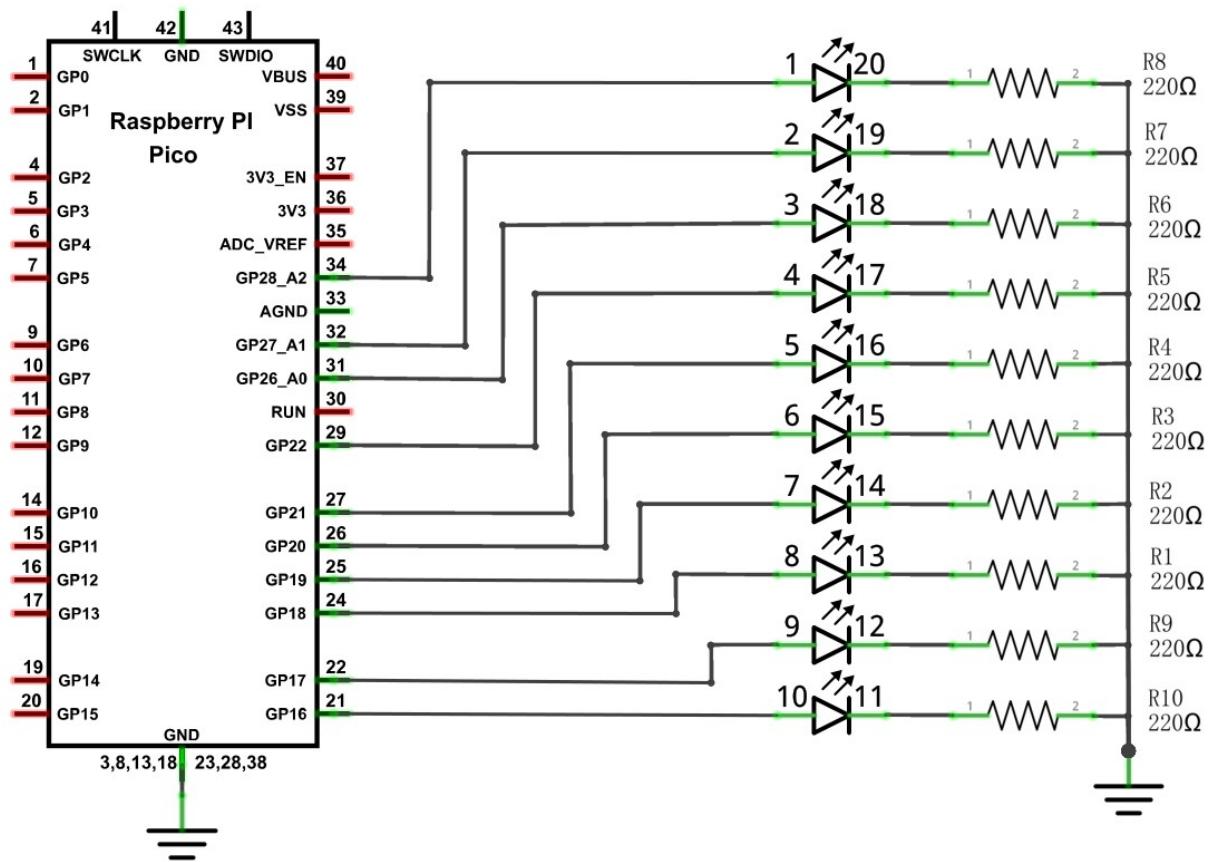
Habiendo aprendido sobre PWM, podemos usarlo para controlar el gráfico de barras LED y realizar una luz que fluye más fresca.

Lista de componentes

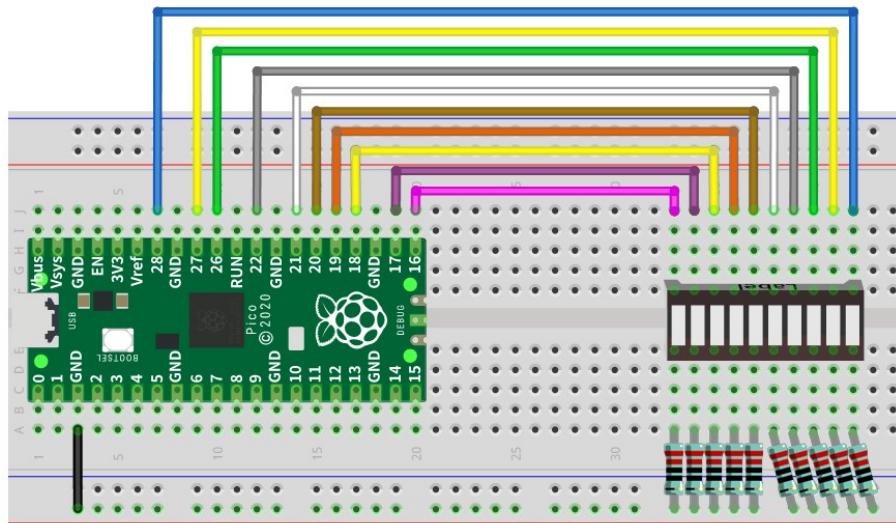
Frambuesa Pi Pico x1	Cable USBx1	
		
Protoboard x1		
Puente	Gráfico de barras LED x1	Resistencia 220Ω x10

Circuito

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



Nota: Para ayudar a los usuarios a tener una mejor experiencia al realizar los proyectos, hemos realizado algunas modificaciones en el diagrama de simulación de Pico. Tenga en cuenta que existen ciertas diferencias entre el diagrama de simulación y la placa real para evitar malentendidos.

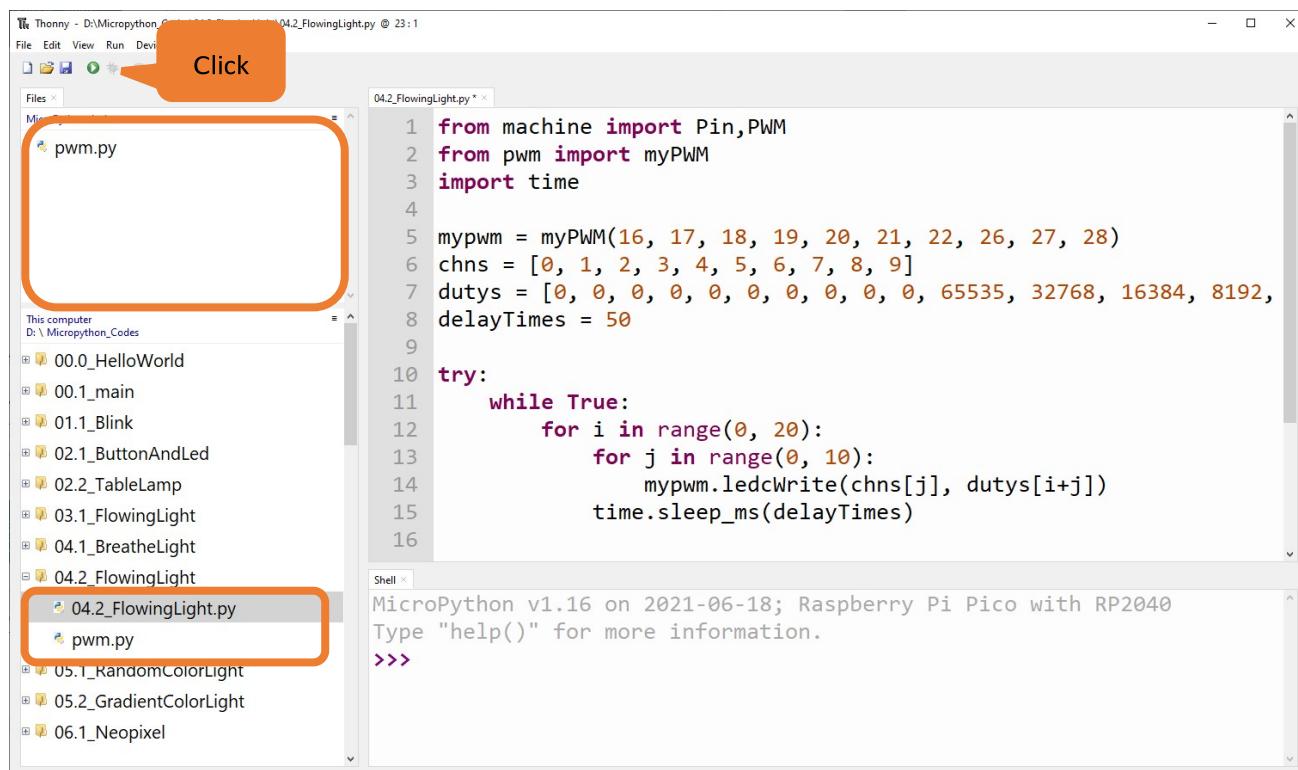
Si LEDbar no funciona, intente girar LEDbar 180°. La etiqueta es aleatoria.

Código

Flowing Light con cola se implementó con PWM.

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "04.2_FlowingLight". Seleccione "pwm.py", haga clic con el botón derecho para seleccionar "Cargar en /", espere a que "pwm.py" se cargue en Raspberry Pi Pico y luego haga doble clic en "04.2_FlowingLight.py".

04.2_FlowingLight



```

from machine import Pin,PWM
from pwm import myPWM
import time

mypwm = myPWM(16, 17, 18, 19, 20, 21, 22, 26, 27, 28)
chns = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
dutys = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 65535, 32768, 16384, 8192,
delayTimes = 50

try:
    while True:
        for i in range(0, 20):
            for j in range(0, 10):
                mypwmyledcWrite(chns[j], dutys[i+j])
            time.sleep_ms(delayTimes)

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar secuencia de comandos actual", y el gráfico de barras LED se iluminará y apagará gradualmente de izquierda a derecha, luego se encenderá y apagará de derecha a izquierda. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa. El siguiente es el código del programa:

1	desde pin de
2	importación de máquina
3	, PWM desde pwm import
4	myPWM import time
5	mypwm = myPWM(16, 17, 18, 19, 20, 21, 22, 26,
6	27, 28) chns = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
7	derechos = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 65535, 32768, 16384, 8192, 4096, 2048, 1024,
8	512, 256,
9	128, 0, 0, 0, 0, 0, 0, 0, 0, 0]
10	tiempos de retraso = 50
11	prueba: mientras que es
12	cierto: para i
	en rango (0, 20):

```
13         para j en el rango (0, 10):
14         mypwm.ledcWrite(chns[j], deberes[i+j]) time.sleep_ms(delayTimes)
15             para i
16             dieciséis en rango (0, 20): para j en
17                 rango (0, 10):
18                     mypwm.ledcWrite(chns[9-j], deberes[i+j])
19                     time.sleep_ms(delayTimes) excepto :
20                         mypwm.deinit()
21
```

Importe el objeto myPWM desde pwm.py y configure los pines correspondientes para el canal PWM.

```
2 desde pwm importar myPWM  
...  
5 mypwm = myPWM(16, 17, 18, 19, 20, 21, 22, 26, 27, 28)
```

Cree un objeto para myPWM y configure 10 pines de salida PWM. Defina 10 canales PWM y 30 valores de ancho de pulso.

```

5 mypwm = myPWM(16, 17, 18, 19, 20, 21, 22, 26, 27, 28) chns = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
6 deberes = [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 65535, 32768, 16384, 8192, 4096, 2048, 1024, 512,
7 256, 128, 0, 0, 0, 0, 0, 0, 0, 0]
```

Llame a ledcWrite() para establecer los deberes del ciclo de trabajo [i+j] para el canal chns[i] de PWM.

14 mypwm. ledcWrite(chns[j], deberes[i+j])

Apague el PWM del objeto myPWM.

22 | mypwm.deinit()

En el código, se utiliza un anidamiento de dos bucles for para lograr este efecto.

```
12         para i en rango (0,
13         20): para j en rango (0,
14         10):
15         mypwm.ledcWrite(chns[j], deberes[i+j]) time.sleep_ms(de
diecisésis           para i
17           en rango (0, 20): para j
18           en rango (0, 10):
19           mypwm.ledcWrite(chns[9 -j], deberes[i+j]) time.sleep_ms
```

En la función principal, se usa un bucle for anidado para controlar el ancho de pulso del PWM. Cada vez que i en el primer bucle for aumenta en 1, el gráfico de barras LED se moverá una cuadrícula y cambiará gradualmente de acuerdo con el valor de los deberes de la matriz. Como se muestra en la siguiente tabla, el valor en la segunda fila es el valor de los deberes de la matriz, y las 10 cuadrículas verdes en cada fila a continuación representan los 10 LED en el gráfico de barras de LED. Cada vez que i aumenta en 1, el valor del gráfico de barras LED se moverá hacia la derecha una cuadrícula y, cuando llegue al final, se moverá desde el final hasta el punto de inicio, logrando el efecto deseado.

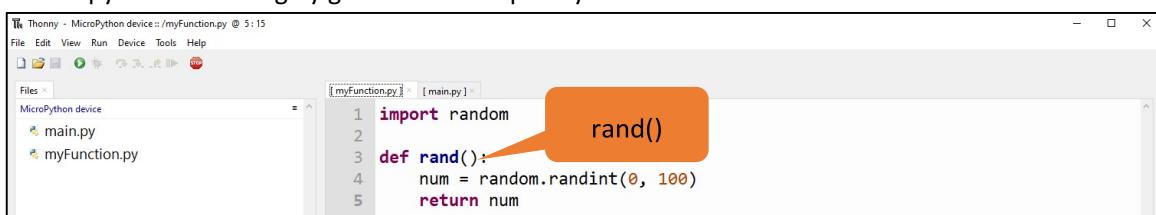
0	1	2	3	...	7	8	9	1	1	1	1	1	1	1	1	1	1	2	2	...	2	2	3			
0	1	2	3	...	7	8	9	1	0	1	2	3	4	5	6	7	8	9	0	1	2	...	2	8	9	0
d	0	0	0	...	0	0	0	0	6	3	1	8	4	2	1	5	2	1	0	0	0	1	5	2	0	
i									5	2	6	1	0	0	0	0	1	5	2	6	8					
									5	7	3	9	9	4	2	2	6	8								
									3	6	8	2	6	8	4											
									5	8	4															
0																										
1																										
...																										
18																										
19																										
20																										

Cómo importar un módulo de python personalizado

Cada archivo de Python, siempre que esté almacenado en el sistema de archivos de Raspberry Pi Pico, es un módulo. Para importar un módulo personalizado, el archivo del módulo debe estar ubicado en la ruta de la variable de entorno de MicroPython o en la misma ruta que el programa que se está ejecutando actualmente.

Código

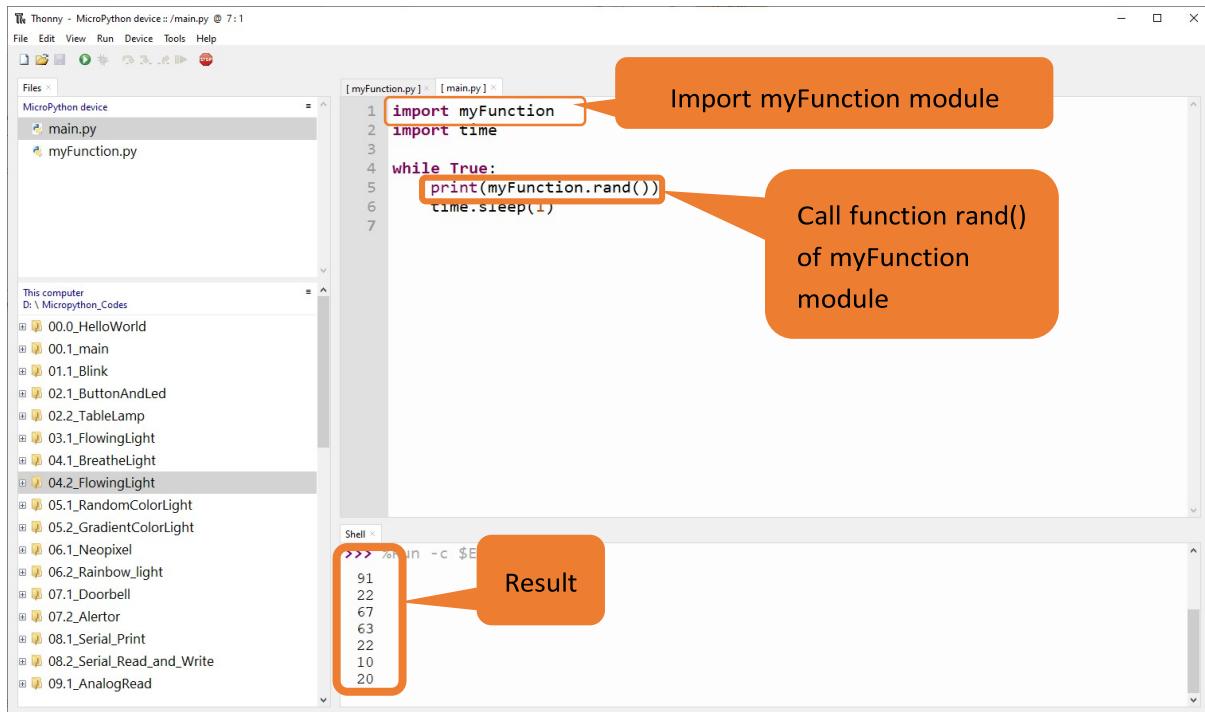
Primero, personalice un módulo de python "myFunction.py". Cree un nuevo archivo py y asígnele el nombre "myFunction.py". Escriba código y guárdelo en Raspberry Pi Pico.



```
thonny - MicroPython device ::/myFunction.py @ 5:15
File Edit View Run Device Tools Help
File Explorer Files MicroPython device
main.py myFunction.py
myFunction.py
1 import random
2
3 def rand():
4     num = random.randint(0, 100)
5     return num
```

La función rand() crea aleatoriamente un número entero que va de 0 a 99.

En segundo lugar, importe el módulo myFunction "myFunction" a main.py.



The following is the program code:

1. myFunction.py

```

1 importar al azar
2
3 definitivamen
4 te rand () :
5 num = random. randint (0, 100) return num

```

Importar módulo aleatorio.

```
1 importar al azar
```

Llame a la función randit() en el módulo aleatorio para generar aleatoriamente un número entero en el rango de 0-99 y asignarlo a la variable num.

```
4 num = random. randint (0, 100)
```

2. principal.py

```

1 Importar tiempo de
2 importación de
3 myFunction
4 mientras
5 que es
6 cierto:
7     imprimir (myFunction. rand ())
8     time. sleep (1)

```

Importe el módulo myFunction "myFunction" a main.py.

```
1 importar mi función
```

Llame a rand() en el módulo myFunction.

```
5     imprimir (miFunción. rand ())
```

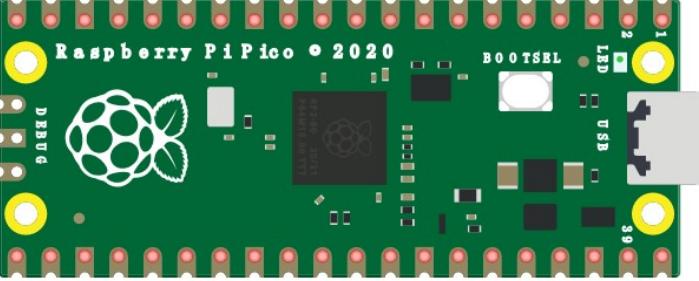
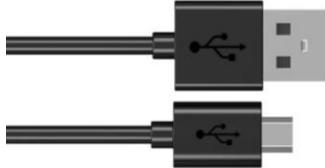
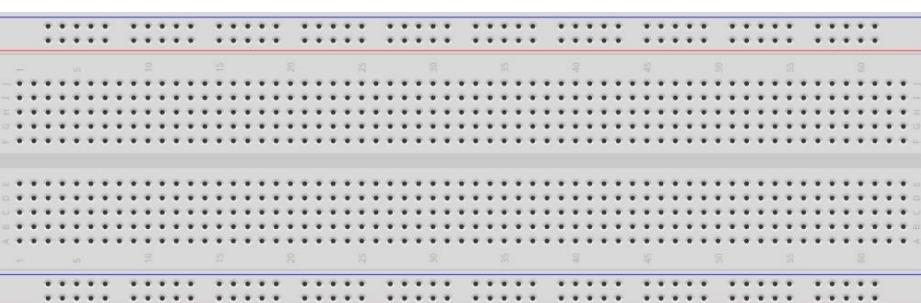

Capítulo 5 LED RGB

En este capítulo, aprenderemos cómo controlar un RGBLED. Puede emitir diferentes colores de luz. A continuación, usaremos RGBLED para hacer una luz multicolor.

Proyecto 5.1 Luz de color aleatorio

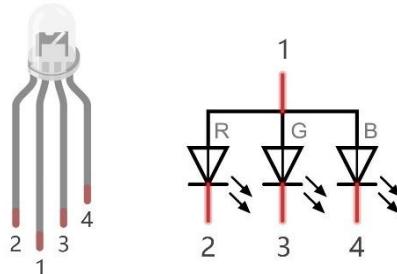
En este proyecto, haremos un LED multicolor. Y podemos controlar RGBLED para cambiar diferentes colores automáticamente.

Lista de componentes

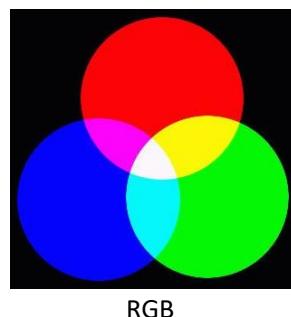
Frambuesa Pi Pico x1	Cable USBx1	
		
Protoboard x1		
		
LED RGB x1	Resistencia 220Ω x3	Puente
		

Conocimiento relacionado

RGB LED tiene 3 LED integrados que pueden emitir respectivamente luz roja, verde y azul. Y tiene 4 pines. El pin largo (1) es el puerto común, es decir, el puerto positivo o negativo de 3 LED. A continuación se muestra el LED RGB con puerto positivo común y su símbolo. Podemos hacer que el LED RGB emita varios colores de luz controlando estos 3 LED para que emitan luz con diferente brillo.



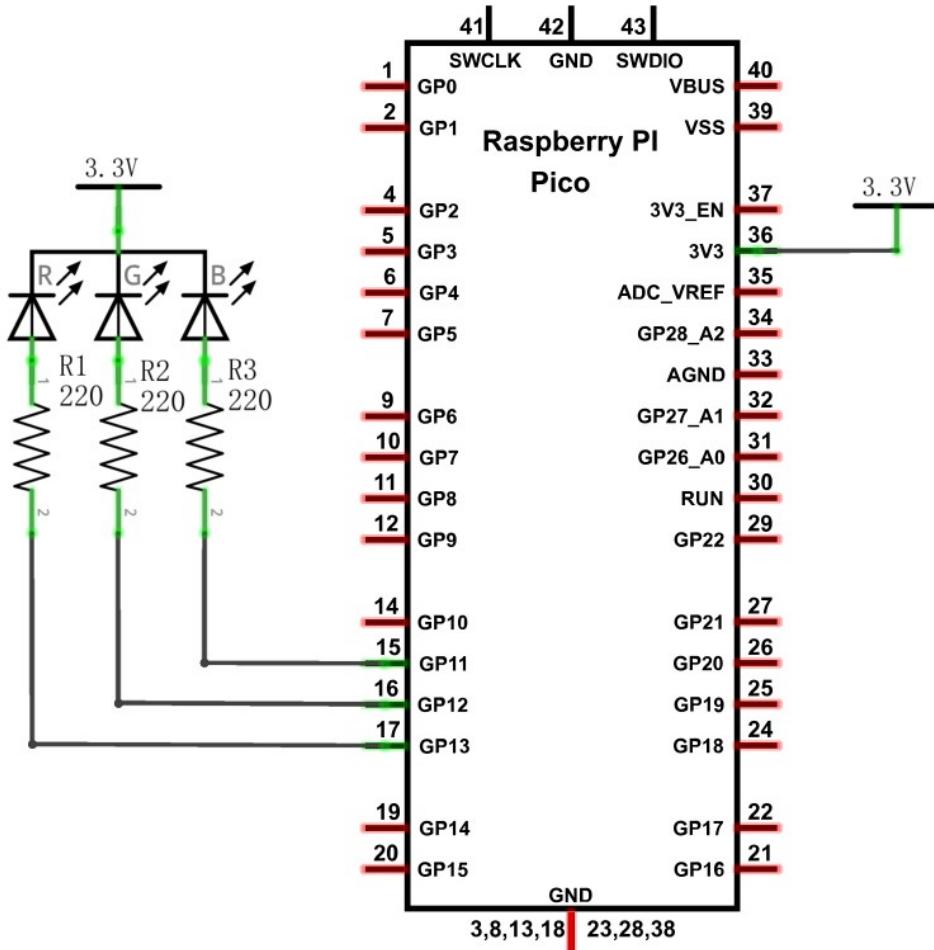
La luz roja, verde y azul se conocen como tres colores primarios. Cuando combina estas tres luces de colores primarios con diferente brillo, puede producir casi todo tipo de luces visibles. Las pantallas de computadora, un solo píxel de la pantalla del teléfono celular, el neón, etc. funcionan bajo este principio.



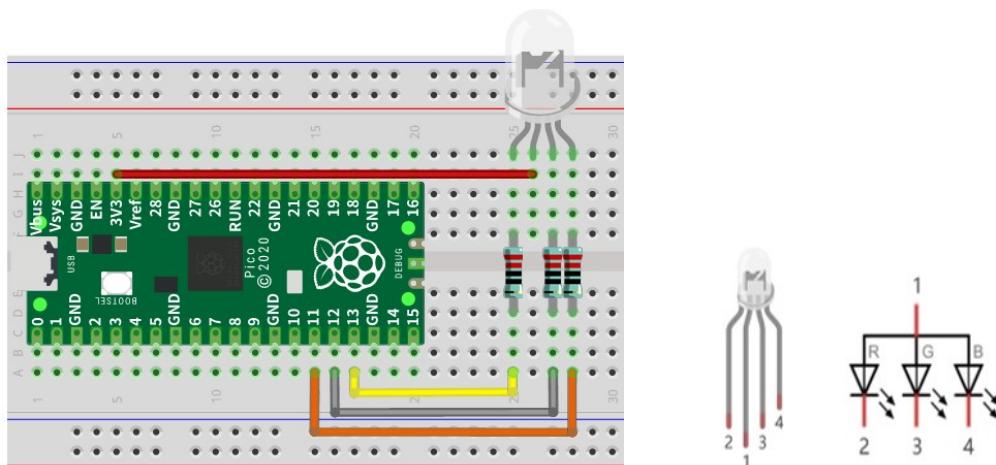
Si usamos tres PWM de 10 bits para controlar el RGBLED, en teoría, podemos crear $2^{10} * 2^{10} * 2^{10} = 1,073,741,824$ (mil millones) de colores a través de diferentes combinaciones.

Circuito

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



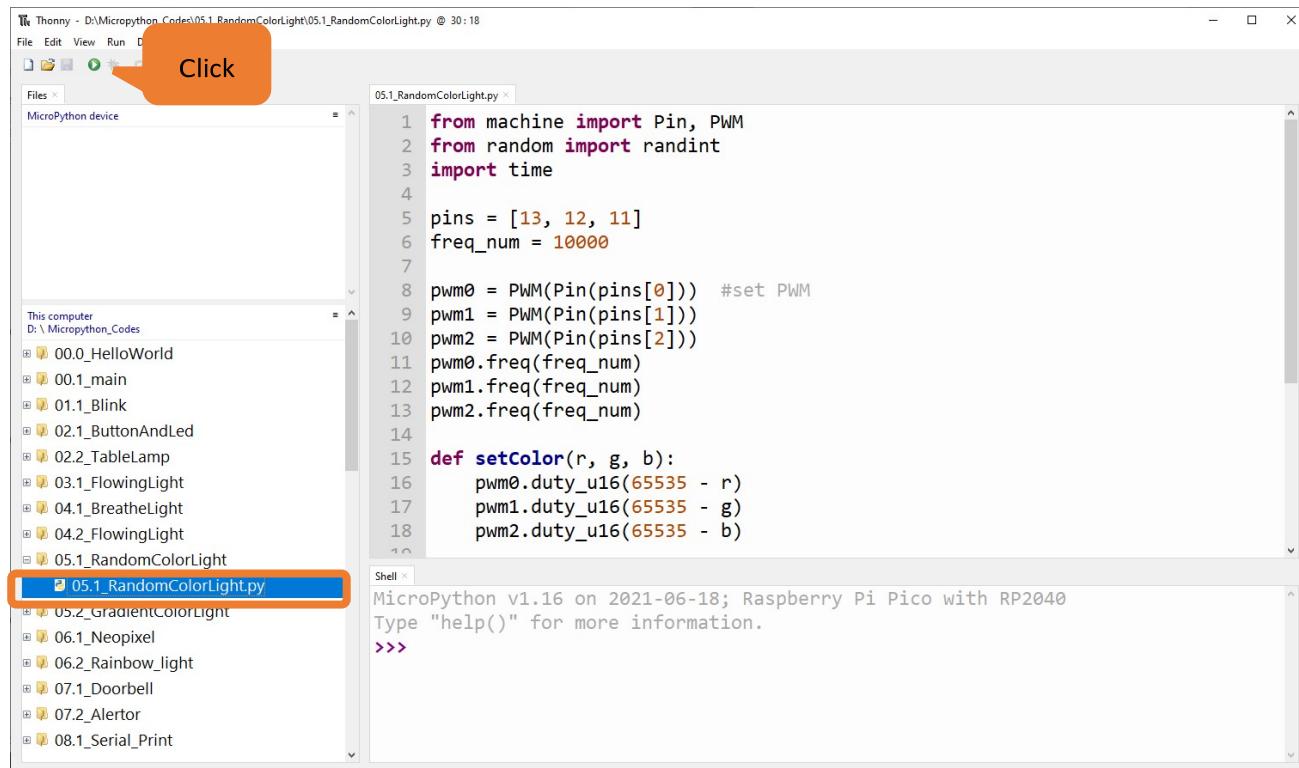
Nota: Para ayudar a los usuarios a tener una mejor experiencia al realizar los proyectos, hemos realizado algunas modificaciones en el diagrama de simulación de Pico. Tenga en cuenta que existen ciertas diferencias entre el diagrama de simulación y la placa real para evitar malentendidos.

Código

Necesitamos crear tres canales PWM y usar un ciclo de trabajo aleatorio para hacer un color RGBLED aleatorio.

Abra “Thonny”, haga clic en “Esta computadora” → “D:” → “Micropython_Codes” → “05.1_RandomColorLight” y haga doble clic en “05.1_RandomColorLight.py”.

05.1_RandomColorLight



```

from machine import Pin, PWM
from random import randint
import time

pins = [13, 12, 11]
freq_num = 10000

pwm0 = PWM(Pin(pins[0])) #set PWM
pwm1 = PWM(Pin(pins[1]))
pwm2 = PWM(Pin(pins[2]))
pwm0.freq(freq_num)
pwm1.freq(freq_num)
pwm2.freq(freq_num)

def setColor(r, g, b):
    pwm0.duty_u16(65535 - r)
    pwm1.duty_u16(65535 - g)
    pwm2.duty_u16(65535 - b)

```

Haga clic en "Ejecutar secuencia de comandos actual", RGBLED comienza a mostrar colores aleatorios. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.

Si usted tiene alguna inquietud, contáctenos a través de: support@freenove.com

El siguiente es el código del programa:

```

1 desde el pin de
2 importación de la máquina
3 , PWM desde el tiempo de
4 importación aleatoria de
5 importación aleatoria
6 pines = [13,
7 12, 11]
8 freq_num =
9 10000
10 pwm0 = PWM(Pin(pines[0]))
11 #establecer PWM pwm1 =
12 PWM(Pin(pines[1])) pwm2 =
13 PWM(Pin(pines[2]))
14 pwm0.freq(freq_num)
15 pwm1.freq(freq_num)
16 pwm2.freq(núm_frecuencia)
17 def
18 establecerColor
19 (r, g, b):
20     pwm0.duty_u16(65535 - r)
21     pwm1.duty_u16(65535 - g)
22     pwm2.duty_u16(65535 - b)

23
24 prueba :
25 mientras
26 que es
27 cierto :
28     rojo = randint(0, 65535)
29     verde = randint(0, 65535)
30     azul = randint(0, 65535)
31     setColor(rojo, verde, azul)
32     time.sleep_ms(200) excepto
33     :
34         pwm0.deinit()
35         pwm1.deinit()
36         pwm2.deinit()

```

Importar módulos de función Pin, PWM y Random.

```

1 desde el pin de importación de la
2 máquina , PWM desde el tiempo de
3 importación aleatoria de
importación aleatoria

```

Configure el modo de salida de GP11, GP12 y GP13 como salida PWM y frecuencia PWM como 10000Hz.

```

5   pines = [13, 12, 11]
6   freq_num = 10000
7
8   pwm0 = PWM(Pin(pines[0]))
9   #establecer PWM pwm1 =
10  PWM(Pin(pines[1])) pwm2 =
11  PWM(Pin(pines[2]))
12  pwm0.freq(freq_num)
13  pwm1.freq(freq_num )
14
15  pwm2.freq(núm_frecuencia)

```

Defina una función para establecer el color de RGBLED.

```

15      def establecerColor (r, g, b):
16          pwm0.duty_u16(65535 - r)
17          pwm1.duty_u16(65535 - g)
18          pwm2.duty_u16(65535 - b)

```

Llame a la función aleatoria randint() para generar un número aleatorio en el rango de 0-65535 y asigne el valor a rojo.

```

22      rojo = aleatorio (0, 65535)

```

Obtenga 3 números aleatorios cada 200 milisegundos y llame a la función setColor para hacer que la pantalla RGBLED muestre colores deslumbrantes.

```

21      mientras que es cierto :
22          rojo = randint(0, 65535) verde =
23              randint(0, 65535) azul = randint(0,
24                  65535) setColor(rojo, verde, azul)
25          time.sleep_ms(200)
26

```

Referencia

Clase aleatoria

Antes de cada uso del módulo.

al azar, agregue la declaración "import random" a la parte superior de Python

randint(inicio, fin)

inicio : fin : Genera aleatoriamente un número entero entre el valor de inicio y final.

aleatorio() : inicial en el rango especificado, que se incluiría en el rango.

aleatorio.unifor . al en el rango especificado, que se incluiría en el rango.

inicio : final : Genera aleatoriamente un número de coma flotante entre 0 y 1. **m(start, end)** : Genera : aleatorio.

Por ejemplo:

tamaño = 4, inicial en el rango especificado, que se incluiría en el rango.

tamaño = 8, al en el rango especificado, que se incluiría en el rango. **getrandbits(tamaño)** : Genera

e incrementar al pas

start : end : Genera un número entero en el rango de 0 a 0b1110.

paso : un : Genera un número entero en el rango de 0 a 0b11111110.

random.seed(sed) : Genera aleatoriamente un entero positivo en el rango de principio a fin generador de números.

sed : Semilla aleatoria en el rango especificado, que se incluiría en el rango.

random.choice(obj) : al en el rango especificado, que se incluiría en el rango.

objeto : lista :ero que especifica el incremento.

de : especifica una semilla aleatoria, que normalmente se aplica junto con otras aleatorias.

, un punto de partida en la generación de números aleatorios.

: Genera aleatoriamente un elemento del objeto obj.

ementos.

Proyecto 5.2 Luz de color degradado

En el proyecto anterior, dominamos el uso de RGBLED, pero la pantalla de colores aleatorios es bastante rígida.

Este proyecto realizará una Luz de moda con suaves cambios de color.

Lista de componentes, el circuito es exactamente el mismo que la luz de color aleatorio del proyecto. Usando un modelo de color, el color cambia de 0 a 255 como se muestra a continuación.



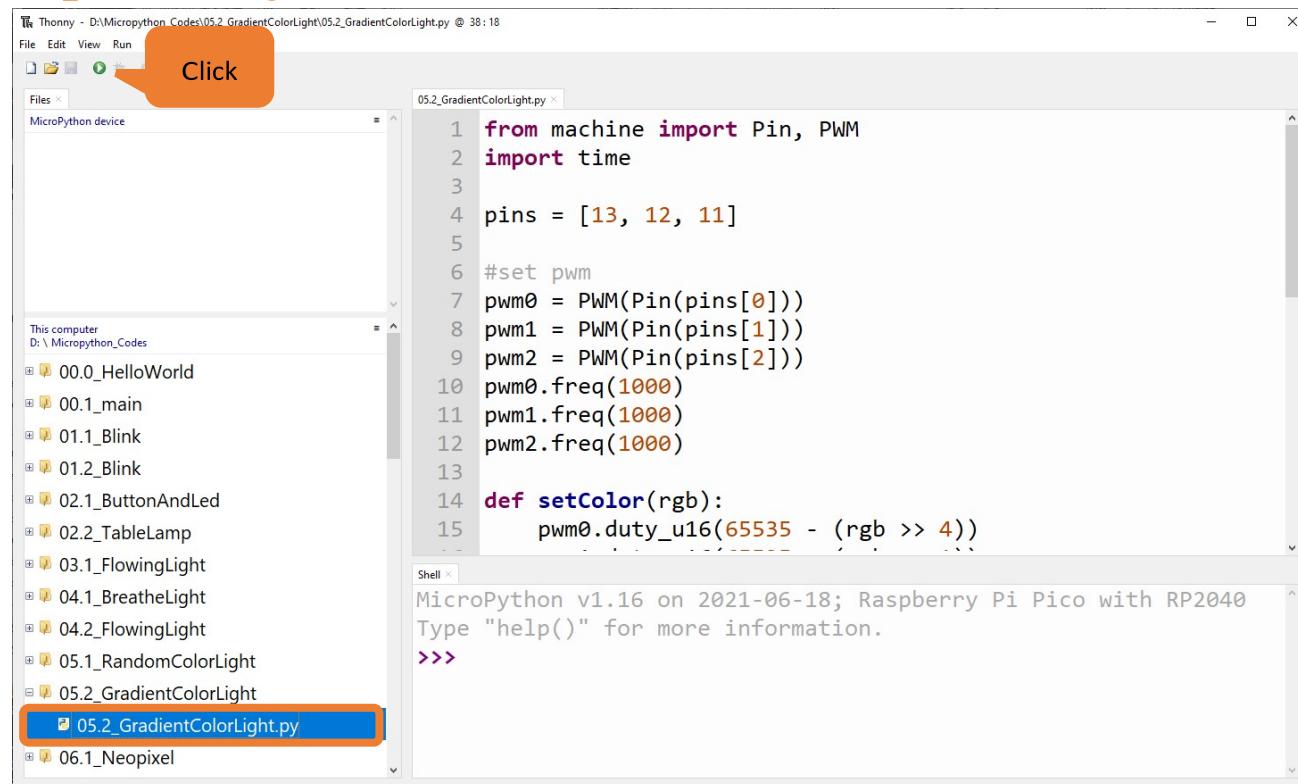
Código

En este código, se implementará el modelo de color y RGBLED cambiará los colores a lo largo del modelo.

¿Cualquier duda?  support@freenove.com

Abra “Thonny”, haga clic en “Esta computadora” → “D:” → “Micropython_Codes” → “05.2_GradientColorLight” y haga doble clic en “05.2_GradientColorLight.py”.

05.2_GradientColorLight



```

from machine import Pin, PWM
import time

pins = [13, 12, 11]

#set pwm
pwm0 = PWM(Pin(pins[0]))
pwm1 = PWM(Pin(pins[1]))
pwm2 = PWM(Pin(pins[2]))
pwm0.freq(1000)
pwm1.freq(1000)
pwm2.freq(1000)

def setColor(rgb):
    pwm0.duty_u16(65535 - (rgb >> 4))

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar secuencia de comandos actual" y la luz emitida por RGBLED cambiará gradualmente. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.

El siguiente es el código del programa:

```
1 importación de máquina ,
2 tiempo de importación de PWM
3 pines = [13,
4     12, 11]
5
6 #set pwm pwm0 =
7 PWM(Pin(pines[0]))
8 pwm1 =
9 PWM(Pin(pines[1]))
10 pwm2 =
11 PWM(Pin(pines[2]))
12 pwm0.freq(1000)
13 pwm1.freq(1000 )
14 pwm2.freq(1000)
15
dieciséi definitivame
s nte
17 establecerCo
18 lor (rgb):
19     pwm0.duty_u16(65535 - (rgb >> 4))
20     pwm1.duty_u16(65535 - (rgb >> 1))
21     pwm2.duty_u16(65535 - (rgb >> 0))
22
23 definitiv
24 amente
25 rueda
26 (posición
27 ):
28 WheelPos = pos % 65535
29 si WheelPos < 21845:
30     return (((65535 - WheelPos*3) << 4) | ((WheelPos*3)
31 << 1)) elif WheelPos >= 21845 y WheelPos < 43690: WheelPos -
32 = 21845 return (((65535 - WheelPos*3) << 1) | (RuedaPos*3))
33 más :
34 WheelPos -= 43690
35     volver (((RuedaPos*3) << 4) | (65535 - RuedaPos*3))
36 intente : while True : for
37 i in range (0, 65535):
38     setColor(wheel(i))
     time.sleep_ms(10) excepto:
     pwm0.deinit()
     pwm1.deinit()
     pwm2.deinit()
```

En la función **setColor()** , usamos una variable para representar el valor de RGB, haciéndolo más conveniente para el paso de parámetros. Como el rango del ciclo de trabajo de PWM es 0-65535, que es 2 a la dieciséis potencia, cuando se divide, el valor de cada canal de color se puede obtener con una simple operación bit a bit.

```
14     definitivamente establecerColor (rgb) :
15         pwm0. duty_u16(65535 - (rgb >> 4))
16         pwm1. duty_u16(65535 - (rgb >> 1))
17         pwm2. duty_u16(65535 - (rgb >> 0))
```

La función **rueda()** es un método de selección de color del modelo de color presentado anteriormente. El rango de valores del parámetro pos es 0-65535. La función devolverá datos que contienen los valores del ciclo de trabajo de 3 pines.

```
19     definitivamente rueda (posición) :
20     WheelPos = pos % 65535 si
21     WheelPos < 21845:
22         return (((65535 - WheelPos*3) << 4) | ((WheelPos*3) << 1))
23     elif WheelPos >= 21845 y WheelPos < 43690: WheelPos -= 21845
24     return (((65535 - WheelPos*3) << 1) | (RuedaPos*3)) más :
25     WheelPos -= 43690 volver (((WheelPos*3) << 4) | (65535 -
26     WheelPos*3))
```

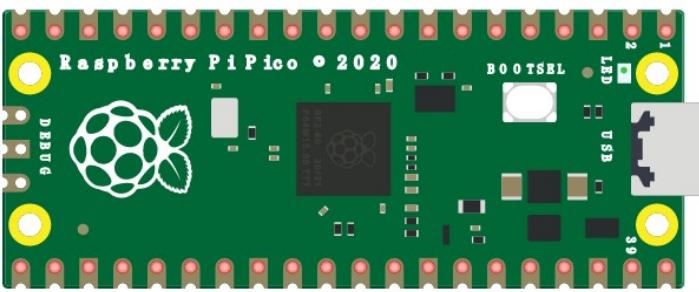
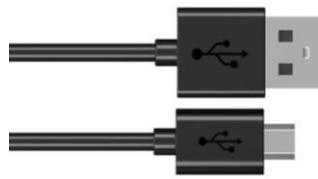
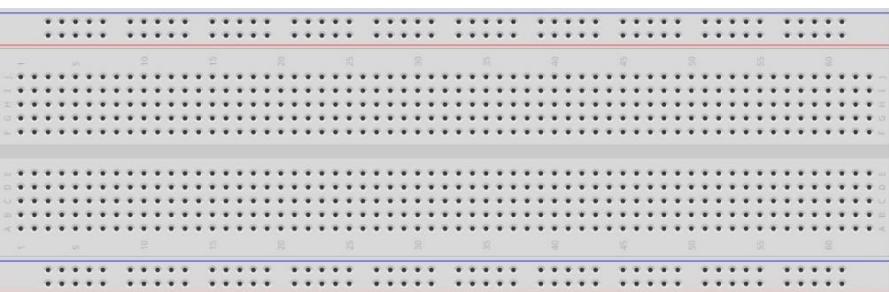
Capítulo 6 Zumbador

En este capítulo, aprenderemos sobre los zumbadores y los sonidos que hacen.

Proyecto 6.1 Timbre

Haremos este tipo de timbre: cuando se presiona el botón, suena el timbre; y cuando se suelta el botón, el zumbador deja de sonar.

Lista de componentes

Frambuesa Pi Pico x1		Cable USBx1	
Protoboard x1			
Puente			
Transistor NPNx1 (S8050)		Zumbador activo x1	
Pulsador x1		Resistor 1kΩ x1	
		Resistor 10kΩ x2	

Capítulo 6 Zumbador

Conocimiento de componentes

Zumbador

El zumbador es un componente de sonido, que se usa ampliamente en dispositivos electrónicos como calculadoras, relojes de advertencia electrónicos y alarmas. El zumbador tiene dos tipos: activo y pasivo. El zumbador activo tiene un oscilador en su interior, que sonará mientras se le suministre energía. El zumbador pasivo requiere una señal de oscilador externo (generalmente usa PWM con una frecuencia diferente) para emitir un sonido.

Zumbador activo Zumbador pasivo



El zumbador activo es fácil de usar. En general, solo puede producir una frecuencia específica de sonido. El zumbador pasivo requiere un circuito externo para emitir un sonido, pero se puede controlar para emitir un sonido con una frecuencia diferente. La frecuencia de resonancia del zumbador pasivo es de 2 kHz, lo que significa que el zumbador pasivo es más fuerte cuando su frecuencia de resonancia es de 2 kHz.

A continuación, usaremos un zumbador activo para hacer un timbre y un zumbador pasivo para hacer una alarma.

¿Cómo identificar el zumbador activo y pasivo?

1. Por lo general, hay una etiqueta en la superficie del zumbador activo que cubre el orificio vocal, pero este no es un método de juicio absoluto.
2. Los zumbadores activos son más complejos que los zumbadores pasivos en su fabricación. Hay muchos circuitos y elementos osciladores de cristal dentro de los zumbadores activos; todo esto generalmente está protegido con una capa impermeable (y una carcasa) que expone solo sus pines desde la parte inferior. Por otro lado, los zumbadores pasivos no tienen revestimientos protectores en la parte inferior. Desde la visualización de los orificios de un zumbador pasivo, puede ver la placa de circuito, las bobinas y un imán permanente (todos o cualquier combinación de estos componentes según el modelo).

Zumbador activo Zumbador pasivo



Transistor

Debido a que el zumbador requiere una corriente tan grande que la capacidad de salida de GP de Raspberry Pi Pico no puede cumplir con el requisito, aquí se necesita un transistor de tipo NPN para amplificar la corriente.

Transistor, el nombre completo: transistor semiconductor, es un dispositivo semiconductor que controla la corriente. El transistor se puede usar para amplificar una señal débil o funciona como un interruptor. Tiene tres electrodos (PINs): base (b), colector (c) y emisor (e). Cuando pasa corriente entre "be", "ce" permitirá que pase varias veces la corriente (aumento del transistor), en este punto, el transistor funciona en el área de amplificación.

Cuando la corriente entre "be" excede un cierto valor, "ce" no permitirá que la corriente aumente más, en este punto, el transistor funciona en el área de saturación. El transistor tiene dos tipos, como se muestra a continuación: PNP y NPN.

Transistor PNP Transistor NPN

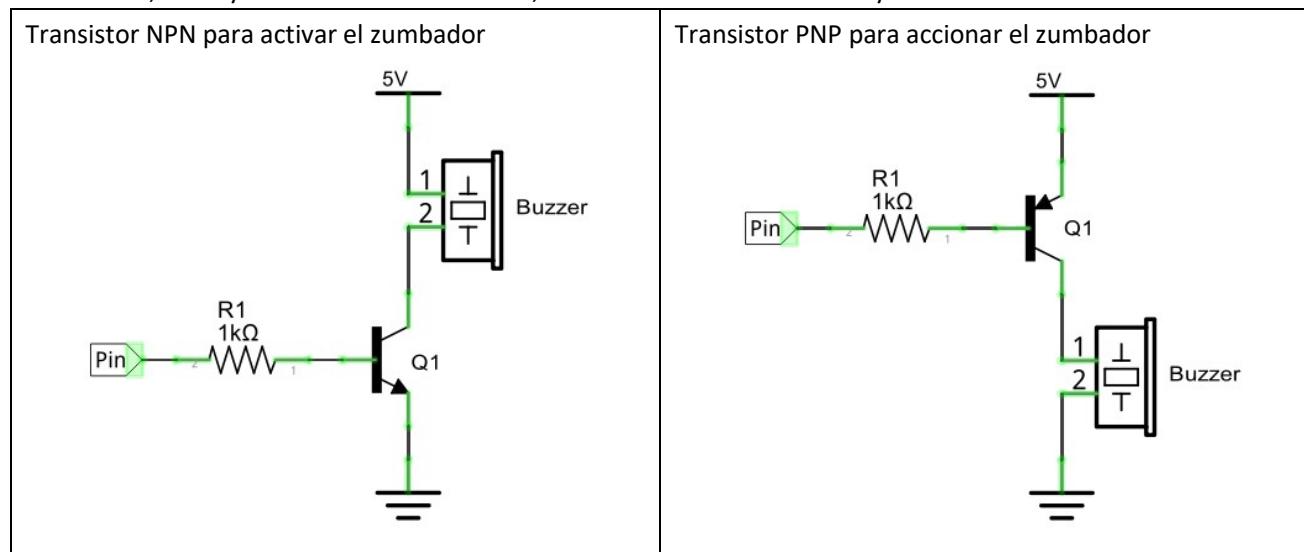


En nuestro kit, el transistor PNP está marcado con 8550 y el transistor NPN está marcado con 8050.

Según las características del transistor, a menudo se usa como interruptor en circuitos digitales. Como la capacidad del microcontrolador para generar corriente es muy débil, usaremos un transistor para amplificar la corriente y controlar los componentes de gran corriente.

Cuando usamos un transistor NPN para controlar el zumbador, a menudo adoptamos el siguiente método. Si GP emite un nivel alto, la corriente fluirá a través de R1, el transistor se conducirá y sonará el zumbador. Si GP emite un nivel bajo, no fluye corriente a través de R1, el transistor no se conducirá y el zumbador no sonará.

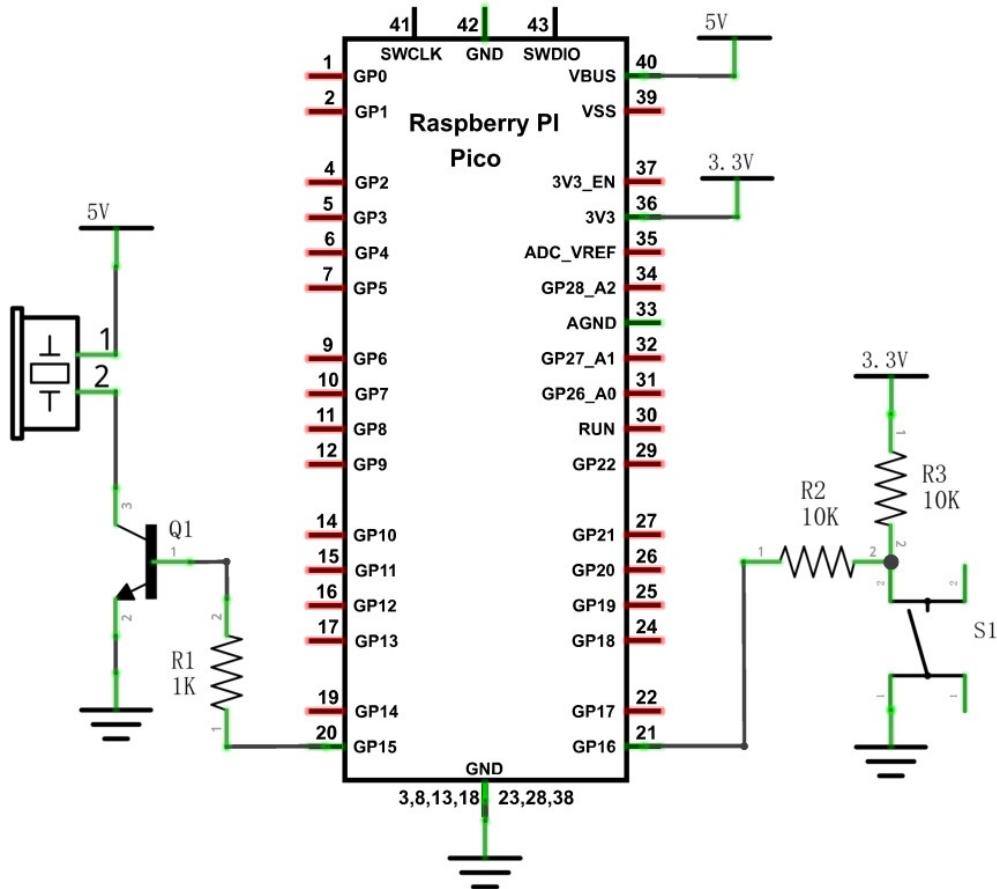
Cuando usamos un transistor PNP para controlar el zumbador, a menudo adoptamos el siguiente método. Si GP emite un nivel bajo, la corriente fluirá a través de R1, el transistor se conducirá y sonará el zumbador. Si GP emite un nivel alto, no fluye corriente a través de R1, el transistor no se conducirá y el zumbador no sonará.



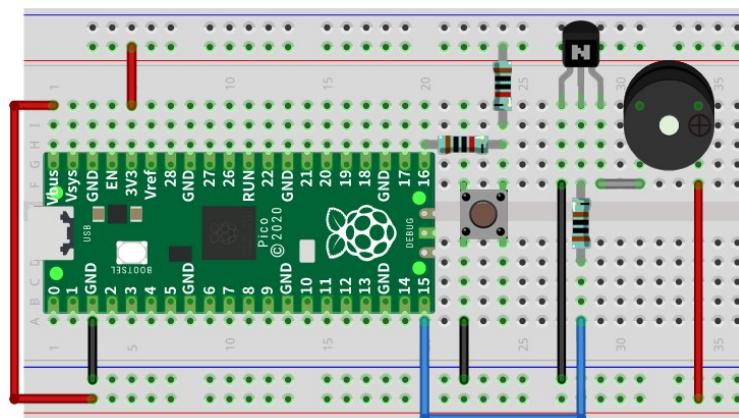
Capítulo 6

Circuito del zumbador

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



Nota:

1. En este circuito, la fuente de alimentación para el zumbador es de 5 V y la resistencia pull-up del botón conectado a la alimentación es de 3,3 V. El zumbador puede funcionar cuando está conectado a una fuente de alimentación de 3,3 V, pero reducirá el volumen.
2. VBUS debe conectarse al extremo positivo del cable USB. Si se conecta a GND, puede quemar la computadora o Raspberry Pi Pico. Del mismo modo, tenga cuidado al cablear los pines 36-40 de Pico para evitar cortocircuitos.

Código

En este proyecto, un zumbador será controlado por un interruptor de botón. Cuando se presiona el interruptor de botón, suena el zumbador y cuando se suelta el botón, el zumbador se detiene. Es análogo a nuestro proyecto anterior que controlaba un LED encendido y apagado.

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "06.1_Doorbell" y haga doble clic en "06.1_Doorbell.py".

06.1_Doorbell

The screenshot shows the Thonny IDE interface. The title bar says "Thonny - D:\Micropython_Codes\06.1_Doorbell\06.1_Doorbell.py @ 12:30". The menu bar includes File, Edit, View, Run, Device, Tools, and Help. The left sidebar is titled "Files" and shows a file tree under "This computer" with "D:\Micropython_Codes" as the root. Inside, there are several projects like "02.1_ButtonAndLed", "02.2_TableLamp", etc., and the "06.1_Doorbell" project is expanded, with "06.1_Doorbell.py" highlighted with an orange rectangle and labeled "Click". The main editor window displays the Python code for "06.1_Doorbell.py":

```
from machine import Pin
import time

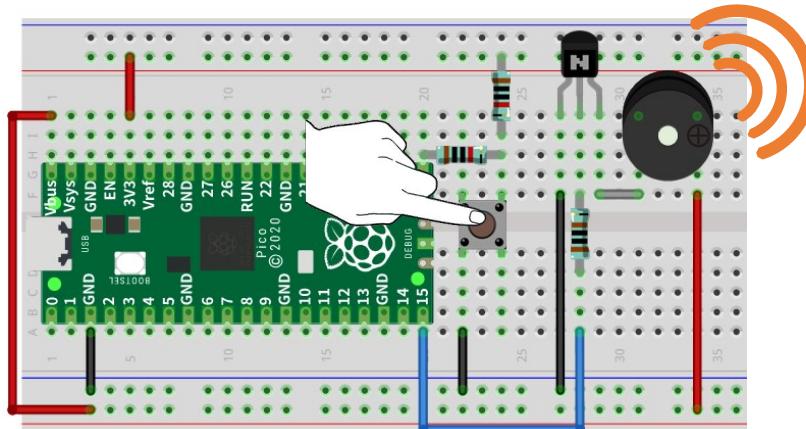
button=Pin(16,Pin.IN,Pin.PULL_UP)
activeBuzzer=Pin(15,Pin.OUT)
activeBuzzer.value(0)

while True:
    if not button.value():
        activeBuzzer.value(1)
    else:
        activeBuzzer.value(0)
```

The bottom shell window shows the MicroPython environment:

```
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

Haga clic en "Ejecutar secuencia de comandos actual", presione el interruptor de botón y sonará el zumbador. Suelte el interruptor de botón y el zumbador se detendrá. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



El siguiente es el código del programa:

```

1 desde la importación de la
2 máquina Tiempo de importación
3 del PIN
4 botón=Pin(16, Pin. IN, Pin. PULL_UP)
5 activeBuzzer=Pin(15, Pin. OUT)
6 activeBuzzer.value(0)
7 while True : si no es
8 button.value() :
9 activeBuzzer.value(1) else :
10 zumbador activo.valor(0)
11
12

```

El código es lógicamente el mismo que usar el botón para controlar el LED.

[Capítulo 6 Zumbador](#)

Proyecto 6.2 Alerta

A continuación, usaremos un zumbador pasivo para hacer una alarma.

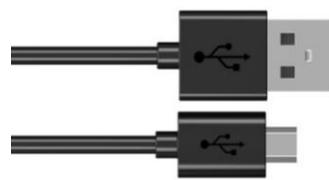
La lista de componentes y la parte del circuito es similar a la última sección. En el circuito del Timbre solo es necesario sustituir el **zumbador activo** por un **zumbador pasivo**.

Lista de componentes

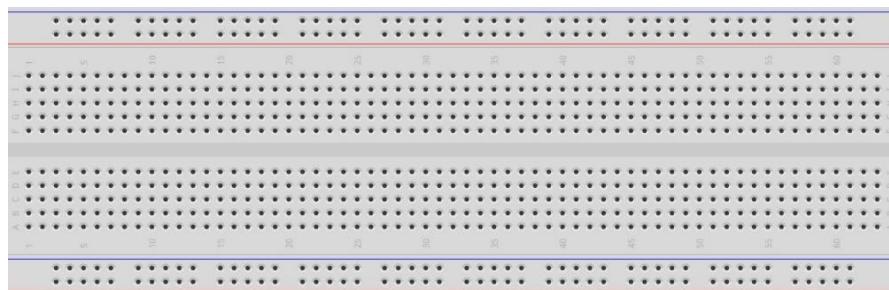
Frambuesa Pi Pico x1



Cable USBx1



Protopboard x1



Puente

Transistor NPNx1
(S8050)

Zumbador pasivo x1

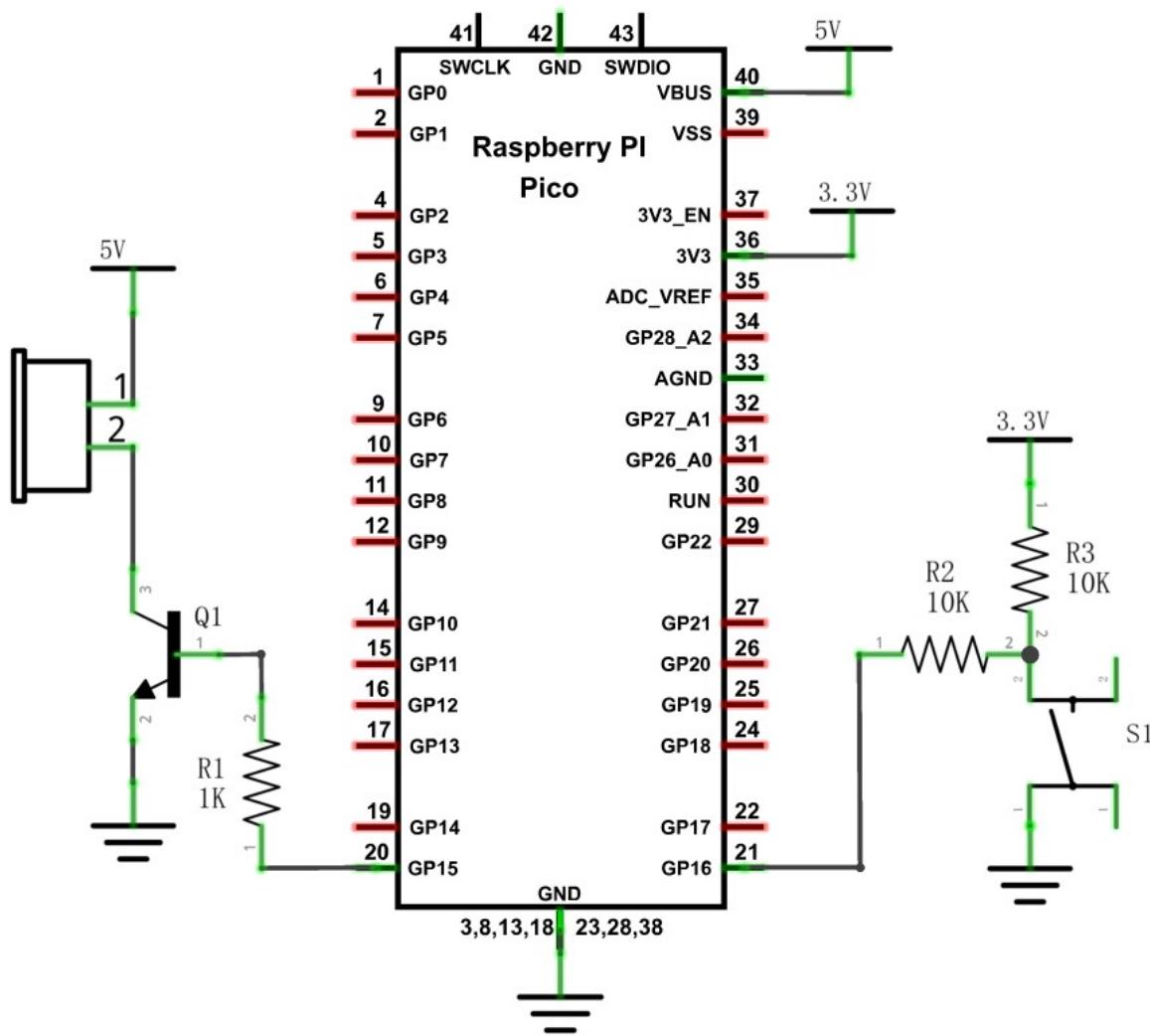


Pulsador x1

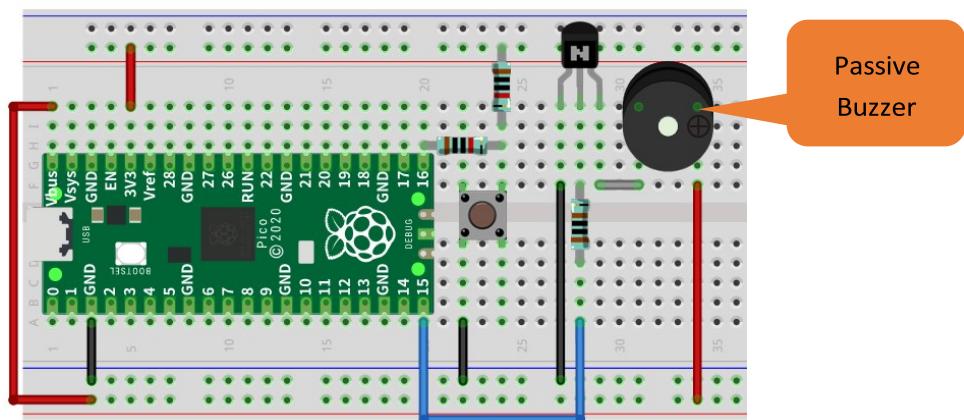
Resistor
1kΩ x1Resistor
10kΩ x2

Circuito

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



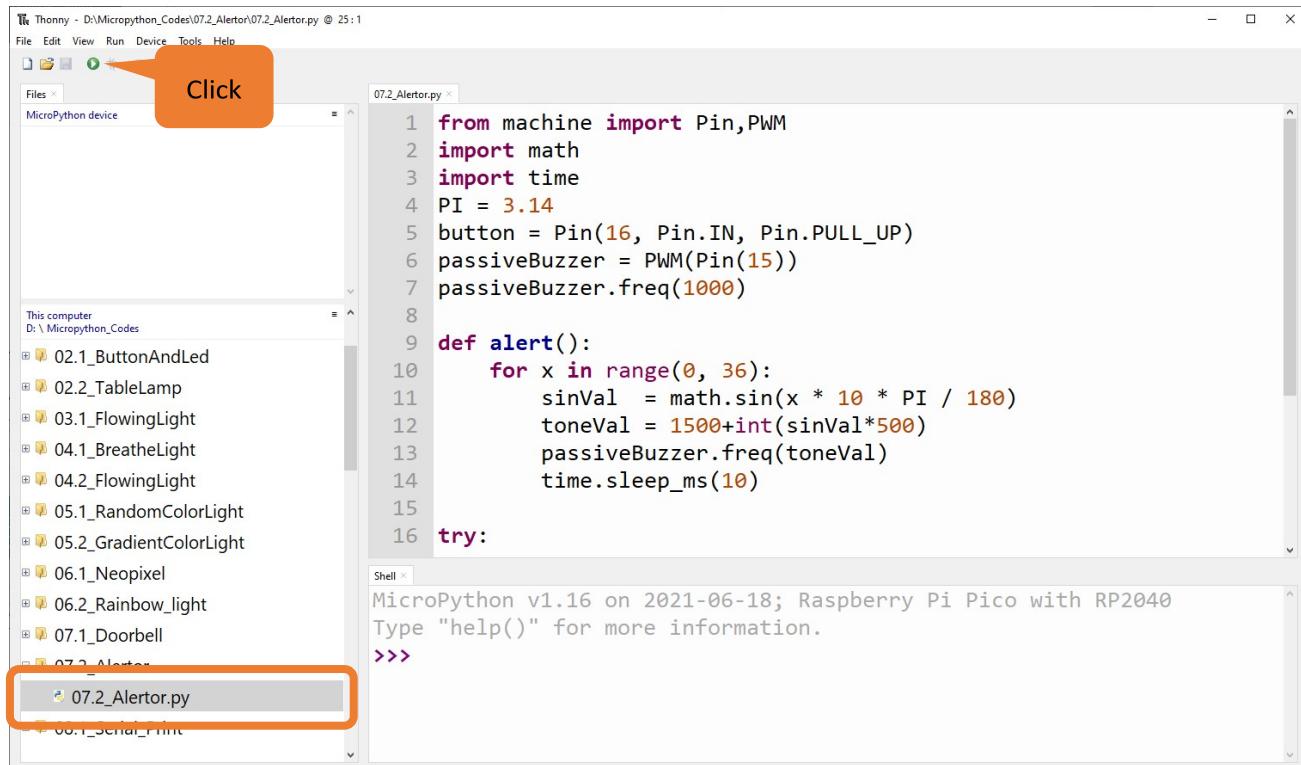
Capítulo 6 Zumbador

Código

En este proyecto, la alarma del zumbador se controla mediante el botón. Presione el botón, luego suena el zumbador. Si suelta el botón, el zumbador dejará de sonar. Lógicamente, es lo mismo que usar el botón para controlar el LED. En cuanto al método de control, el zumbador pasivo requiere PWM de cierta frecuencia para sonar.

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "06.2_Alertor", y haga doble clic en "06.2_Alertor.py".

06.2_Alertor



```

from machine import Pin,PWM
import math
import time
PI = 3.14
button = Pin(16, Pin.IN, Pin.PULL_UP)
passiveBuzzer = PWM(Pin(15))
passiveBuzzer.freq(1000)

def alert():
    for x in range(0, 36):
        sinVal = math.sin(x * 10 * PI / 180)
        toneVal = 1500+int(sinVal*500)
        passiveBuzzer.freq(toneVal)
        time.sleep_ms(10)

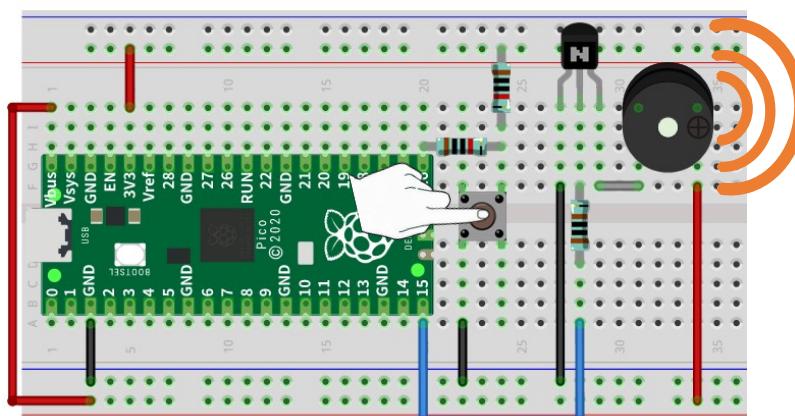
try:
    while True:
        if not button.value():
            alert()
        else:
            break

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar script actual", presione el botón y sonará la alarma. Y cuando se suelta el botón, la alarma dejará de sonar. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.

Si el sonido del zumbador es demasiado alto o demasiado bajo para usted, puede modificar el ciclo de trabajo o la frecuencia de PWM.



El siguiente es el código del programa:

¿Cualquier duda?  support@freenove.com

```

1   desde máquina importar Pin,
2   PWM importar matemáticas
3   tiempo de importación PI =
4   3.14 botón = Pin(16, Pin.IN,
5   Pin.PULL_UP) zumbador pasivo =
6   PWM(Pin(15)) zumbador
7   pasivo.freq(1000)
8   definitivamente
9   alerta (): para x
10  en rango (0, 36):
11  sinVal = matemáticas.sin(x * 10 * PI / 180)
12  toneVal = 1500 + int(sinVal*500) zumbador
13  pasivo.freq(toneVal) time.sleep_ms(10)
14  intente : while True :
15  si no button.value():
16  pasivaBuzzer.duty_u16(4092*2)
17  alerta() más :
18  pasivoBuzzer.duty_u16(0) excepto :
19  zumbador pasivo.deinit()
20
21
22
23
24

```

Importe módulos PWM, Pin, matemáticos y de tiempo.

```

1   desde pin de importación de
2   máquina , tiempo de
3   importación matemática de
4   importación PWM

```

Defina los pines del botón y del zumbador pasivo.

```

4   PI = 3.14 botón = Pin(16, Pin.IN,
5   Pin.PULL_UP) zumbador pasivo =
6   PWM(Pin(15)) zumbador
7   pasivo.freq(1000)

```

Llame a la función sin del módulo matemático para crear los datos de frecuencia del zumbador pasivo.

```

9   definitivamente alerta () :
10  para x en rango (0, 36):
11  sinVal = matemáticas.sin(x * 10 * PI / 180) toneVal =
12  1500 + int(sinVal*500) zumbador pasivo.freq(toneVal)
13  time.sleep_ms(10)
14

```

Cuando no use PWM, apáguelo a tiempo.

```

24  zumbador pasivo.deinit()

```

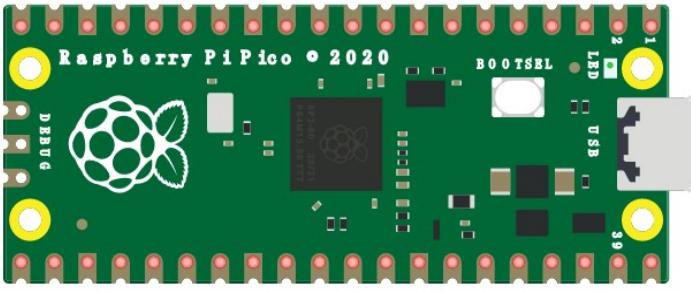
Capítulo 7 Comunicación en serie

La comunicación serial es un medio de comunicación entre diferentes dispositivos. Esta sección describe la comunicación en serie de Raspberry Pi Pico.

Proyecto 7.1 Impresión en serie

Este proyecto utiliza el comunicador serial Raspberry Pi Pico para enviar datos a la computadora e imprimirlos en el monitor serial.

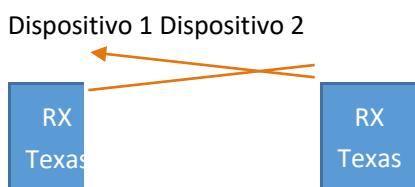
Lista de componentes

Frambuesa Pi Pico x1	Cable USBx1
	

Conocimiento relacionado

Comunicación serial

La comunicación en serie generalmente se refiere al receptor/transmisor asíncrono universal (UART), que se usa comúnmente en la comunicación de circuitos electrónicos. Tiene dos líneas de comunicación, una se encarga de enviar datos (línea TX) y la otra de recibir datos (línea RX). Las conexiones de comunicación serie que utilizan dos dispositivos son las siguientes:



Antes de que comience la comunicación en serie, la velocidad en baudios de ambos lados debe ser la misma. La comunicación entre dispositivos solo puede funcionar si se utiliza la misma velocidad en baudios. Las velocidades de transmisión comúnmente utilizadas son 9600 y 115200.

Puerto serie en Raspberry Pi Pico

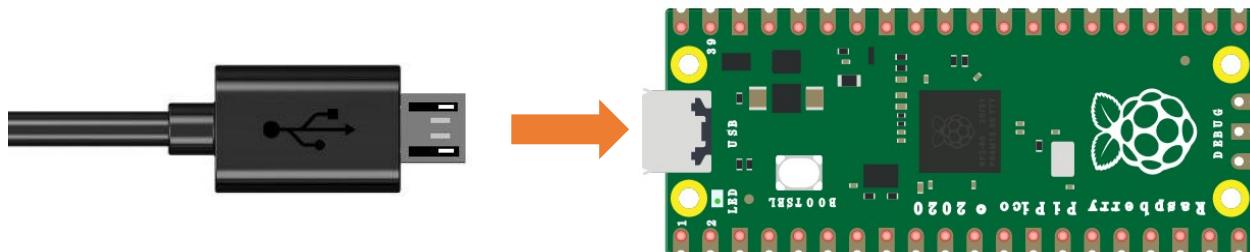
Raspberry Pi Pico ha integrado la transferencia de USB a serie, por lo que podría comunicarse con la computadora que se conecta al cable USB.

Pico USB a computadora serie



Circuito

Conecte Raspberry Pi Pico a la computadora con un cable USB.



Código

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "07.1_Serial_Print" y doble "07.1_Serial_Print.py".

07.1_Serial_Print

```

import time

print("Raspberry Pi Pico initialization completed !")

while True:
    print("Running time : ", time.time()%60,"s")
    time.sleep(1)
  
```

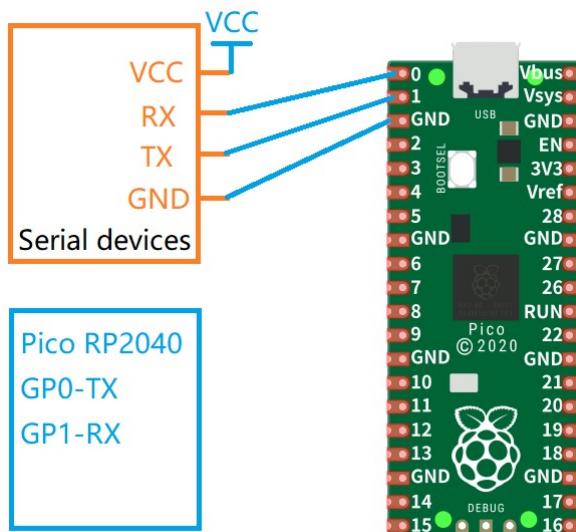
Haga clic en "Ejecutar secuencia de comandos actual" y observe los cambios de "Shell", que mostrará la hora en que Raspberry Pi Pico se enciende una vez por segundo.

```
Shell ×
Raspberry Pi Pico initialization completed !
Running time : 22 s
Running time : 23 s
Running time : 24 s
Running time : 25 s
Running time : 26 s
```

El siguiente es el código del programa:

```
1 tiempo de importación
2
3 imprimir ("¡Se completó la inicialización de Raspberry Pi Pico!")
4 mientras
5 que es
6 cierto:
7     print ("Tiempo de ejecución: ", time.time()%60,"s")
    time.sleep(1)
```

Hay dos comunicaciones seriales en Raspberry Pi Pico: UART0 y UART1. Puede usarlos para comunicarse con dispositivos seriales.



Pin predeterminado para UART0

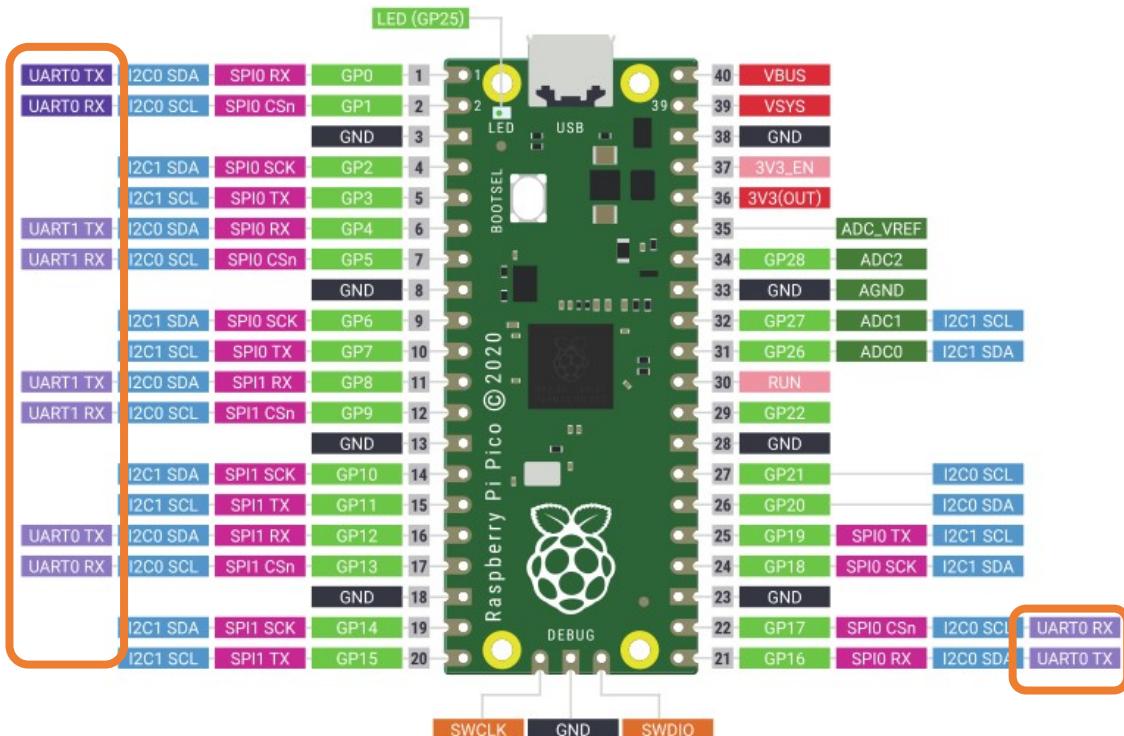
UART0_TX	Alfiler 0
UART0_RX	Alfiler 1

Pin predeterminado para UART1

UART1_TX	Alfiler 4
UART1_RX	Alfiler 5

Para obtener más información, consulte [el pin predeterminado UART, I2C, SPI](#).

Y también puede cambiar la configuración según la distribución de pines.



Referencia

Clase UART

Antes de cada uso del módulo **UART**, agregue la declaración "desde la importación de la máquina **UART**" en la parte superior del archivo python:

a ellos. **id** : tasa de baudios, **bits**, **parity**, **rx**, **tx**, **stop**, **timeout**) : defina puertos serie y configure parámetros para

baudios: bit:

paridad : rx, tx : Puerto de serie. El número de puerto serie disponible es 0

parad: 0 o 1 . Tasa de baudios.

se acabó el tiempo: número de cada carácter.

'verifica pares o impares, con 0 para verificación par y 1 para verificación impar.

UART.leer(nbytes): Pines de lectura y escritura de APT, GP0, GP1, GP4, GP5, GP8, GP9, GP12, GP13, GP16,

UART.leer(): 17. El número de bits de parada, y el bit de parada es 1 o 2.

UART.escribir(buf): período de tiempo de espera (Unidad: milisegundos).

UART.re: 0 < tiempo de espera ≤ 0xFFFF FFFF (decimal: 0 < tiempo de

UART.readii: espera ≤ 2147483647). : Leer nbytes bytes.

: Leer datos.

UART.any() : : Escriba el búfer de bytes en el bus **UART**.

De lo contrario, de : Lee una línea de datos, terminando con un carácter de nueva línea.

0. : Leer y escribir datos en el búfer.

o(buf, nbytes) : Leer y escribir datos en el búfer.

Determine si hay datos en los puertos serie. Si lo hay, devuelve el número de bytes;

Nota: Cuando se usa usART0 o USART1, debe usarse con la placa adaptadora de puerto serie o el dispositivo de puerto serie. De lo contrario, es posible que no observe ningún síntoma.

Proyecto 7.2 Lectura y escritura en serie

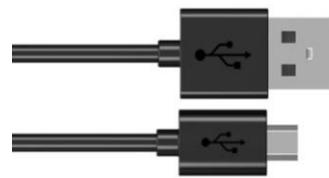
En el siguiente ejemplo, usamos el UART1 de Raspberry Pi Pico para enviar datos a UART0, y leemos los datos recibidos por UART0 e imprimimos a través del "Shell".

Lista de componentes

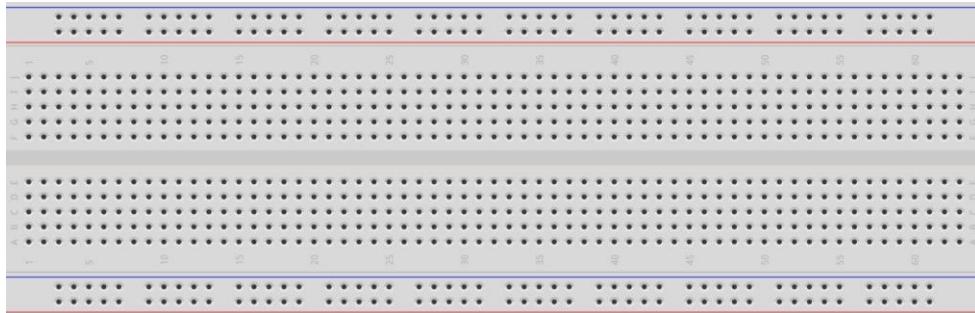
Frambuesa Pi Pico x1



Cable USBx1



Protopboard x1

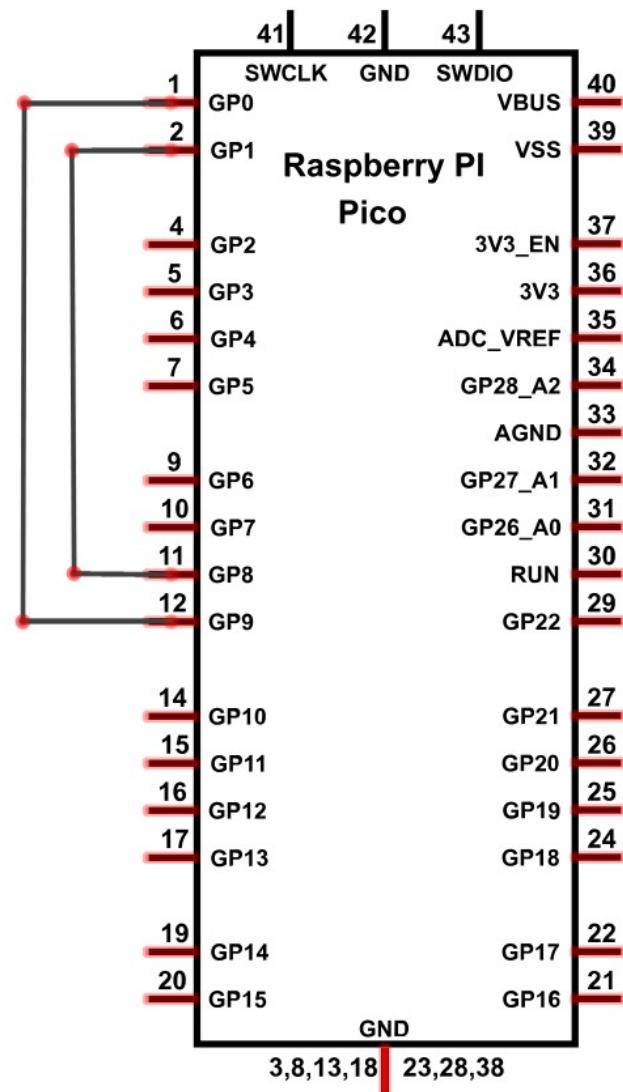


Puente

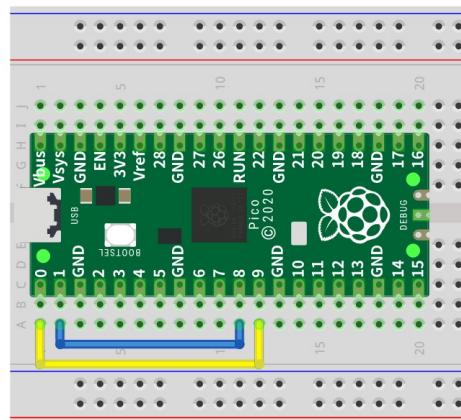


Circuito

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



Código

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "07.2_Serial_Read_and_Write" y haga doble clic en "07.2_Serial_Read_and_Write_UART1_to_UART0.py".

07.2_Serial_Read_and_Write_UART1_to_UART0

```

from machine import UART, Pin
import time

myUsart0 = UART(0, baudrate=9600, bits=8, tx=Pin(0), rx=Pin(1), timeout=10)
myUsart1 = UART(1, baudrate=9600, bits=8, tx=Pin(8), rx=Pin(9), timeout=10)

while True:
    rxData = bytes()
    input_cnt = str(input("myUsart1: "))
    myUsart1.write(input_cnt)
    time.sleep(0.1)
    while myUsart0.any() > 0:
        rxData += myUsart0.read(1)
    print("myUsart0: ", rxData.decode('utf-8'))

```

Haga clic en "Ejecutar script actual". Los usuarios pueden ingresar cualquier dato en "Shell" y presionar Enter. Los datos de entrada se escribirán en UART1 y se enviarán a UART0 para leerlos e imprimirlas en "Shell". Presione "Ctrl+C" o haga clic en "Detener/Reiniciar backend" para salir del programa.

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CODE
myUsart1: 4252
myUsart0: 4252
myUsart1: 415
myUsart0: 415
myUsart1: 5275275
myUsart0: 5275275
myUsart1: 5757857
myUsart0: 5757857
myUsart1: 752752752
myUsart0: 752752752
myUsart1: 444asdfasd
myUsart0: 444asdfasd
myUsart1: dfadfa
myUsart0: dfadfa
myUsart1:

```

El siguiente es el código del programa:

```

1  UART de importación de máquina ,
2  tiempo de importación de PIN
3  myUsart0 = UART(0, velocidad en baudios=9600, bits=8, tx=Pin(0),
4  rx=Pin(1), tiempo de espera=10) myUsart1 = UART(1, velocidad en
5  baudios=9600, bits=8, tx=Pin( 8), rx=Pin(9), tiempo de espera=10)
6  mientras
7  que es
8  cierto :
9  rxData = bytes () input_cnt = str
10 (input("myUsart1: "))
11 myUsart1.write(input_cnt)
12 time.sleep(0.1) while myUsart0.any() >
13 0:
14 rxData += myUsart0.read(1) print ("myUsart0: " ,
rxData.decode('utf-8'))

```

Importe módulos UART, Pin y tiempo.

```

1  UART de importación de máquina ,
2  tiempo de importación de PIN

```

Cree dos objetos UART y configúrelos como los parámetros de UART0 y UART1.

```

4  myUsart0 = UART(0, velocidad en baudios=9600, bits=8, tx=Pin(0), rx=Pin(1), tiempo
5  de espera=10) myUsart1 = UART(1, velocidad en baudios=9600, bits=8, tx=Pin( 8),
rx=Pin(9), tiempo de espera=10)

```

Defina un valor de bytes y asígnelo a rxDate.

```

8  rxData = bytes ()

```

Defina input_cnt para recibir la entrada del usuario y convertirla a un formato de cadena.

```

9  input_cnt = str (entrada("myUsart1: "))

```

myUsart1 llama a la función write() y escribe la entrada del usuario en UART1.

```

10 myUsart1.write(entrada_cnt)

```

myUsart0 llama a la función read() para leer bit a bit los datos enviados por UART1 y guardarlos en rxData. Cuando myUsart0 llama a any() para determinar si UART0 ha leído los datos, cuando any() devuelve 0, UART0 ha leído los datos enviados por UART1.

```

12  mientras myUsart0.any() > 0:
13  rxData += myUsart0.read(1)

```

Se llama a la función decode() para decodificar los datos e imprimirlos en "Shell".

```

14  imprimir ("myUsart0: ", rxData.decode('utf-8'))

```

Capítulo 8 Convertidor AD

En este capítulo aprendemos a usar la función ADC de Raspberry Pi Pico.

Proyecto 8.1 Leer el voltaje del potenciómetro

En este capítulo, usamos la función ADC de Pico para leer la salida de voltaje por potenciómetro.

Lista de componentes

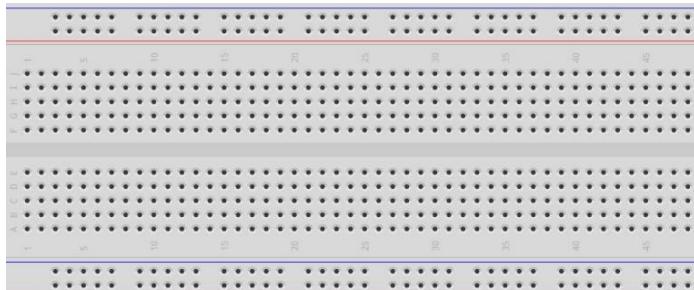
Frambuesa Pi Pico x1



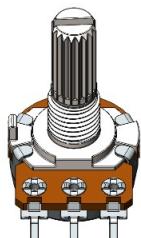
Cable USBx1



Protoboard x1



Potenciómetro giratorio x1



Puente



Conocimiento relacionado

ADC

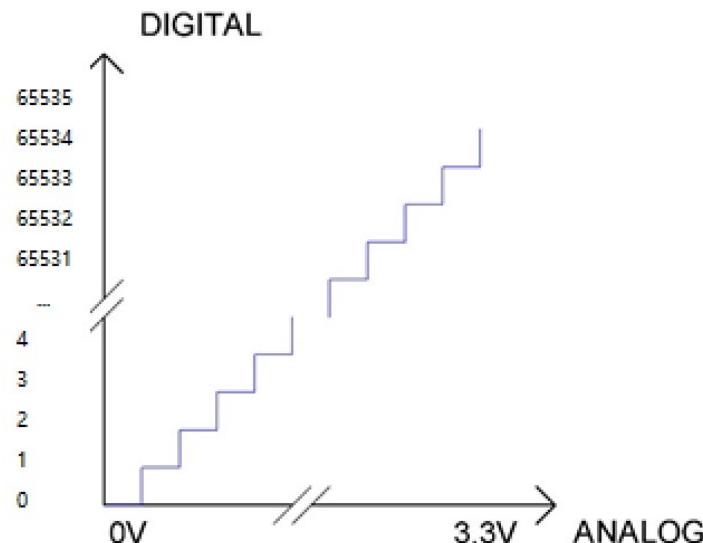
Un convertidor de analógico a digital (ADC) convierte una señal analógica medida en un código digital. ADC tiene dos características clave: resolución y canales.

¿Cualquier duda? support@freenove.com

Resolución ADC de Raspberry Pi Pico

Raspberry Pi Pico utiliza el chip RP2040. Con una resolución ADC de 12 bits, puede convertir cualquier señal analógica a señal digital, en un rango de 0-4095. Por ejemplo, si el rango de voltaje analógico que mide es 0-3.3V, el ADC puede dividirlo en 4096 partes iguales.

Sin embargo, cuando se usa el firmware de Micropython para llamar a Raspberry Pi Pico ADC, la señal digital que obtiene varía de 0 a 65535. Esto se debe a que Micropython procesa internamente la resolución de ADC de Pico a 16 bits, y los valores también se cambian a 0-65535, por lo que para hacerlo igual que el ADC de otros microcontroladores Micropython. La forma en que ADC convierte no cambia, solo cambia la resolución. Entonces, si el voltaje analógico medido oscila entre 0 y 3,3 V, el ADC puede dividirlo en 65536 partes iguales.



Subsección 1: el analógico en un rango de 0V---3.3/65535 V corresponde al 0 digital;

Subsección 2: el analógico en un rango de 3,3/65535 V---2*3,3/65535 V corresponde al digital 1; ...

Subsección 65535: el analógico en el rango de 65534*3,3/65535 V---65535*3,3/65535 V corresponde al digital 65534;

El siguiente análogo se dividirá en consecuencia. La

fórmula de conversión es la siguiente:

Voltaje analógico

$$ADCValue = \frac{\text{_____} * 65535}{3.3}$$

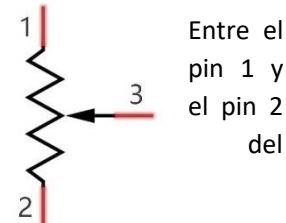
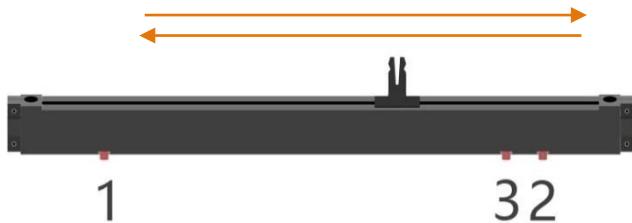
Canales ADC Raspberry Pi Pico

Raspberry Pi Pico tiene 5 canales ADC, que son ADC0 (GP26), ADC1 (GP27), ADC2 (GP28), ADC3 (GP29): se usa para medir VSYS en la placa Pico y ADC4, que se conecta directamente a la temperatura integrada detector de microprocesador RP2040. Por lo tanto, solo hay tres canales ADC genéricos que se pueden usar directamente, a saber, ADC0, ADC1 y ADC2.

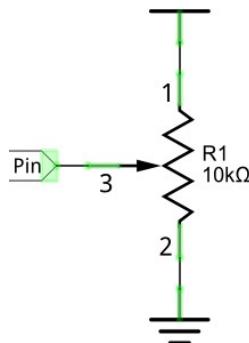
Conocimiento de componentes

Potenciómetro

El potenciómetro es un elemento resistivo con tres partes terminales. A diferencia de las resistencias que hemos usado hasta ahora en nuestro proyecto que tienen un valor de resistencia fijo, el valor de resistencia de un potenciómetro se puede ajustar. Un potenciómetro a menudo está compuesto por una sustancia resistiva (un alambre o elemento de carbón) y un cepillo de contacto móvil. Cuando la escobilla se mueva a lo largo del elemento resistor, habrá un cambio en la resistencia del lado de salida del potenciómetro (3) (o un cambio en el voltaje del circuito que forma parte). La siguiente ilustración representa un potenciómetro deslizante lineal y su símbolo electrónico a la derecha.



Entre el pin 1 y el pin 2 del potenciómetro está el elemento resistivo (un cable de resistencia o carbón) y el pin 3 está conectado a la escobilla que hace contacto con el elemento resistivo. En nuestra ilustración, cuando la escobilla se mueve del pin 1 al pin 2, el valor de resistencia entre el pin 1 y el pin 3 aumentará linealmente (hasta alcanzar el valor más alto del elemento resistivo) y al mismo tiempo la resistencia entre el pin 2 y el el pin 3 disminuirá linealmente y viceversa hasta cero. En el punto medio del control deslizante, los valores de resistencia medidos entre los pines 1 y 3 y entre los pines 2 y 3 serán los mismos. En un circuito, ambos lados del elemento resistivo a menudo están conectados a los electrodos de potencia positivo y negativo. Cuando desliza el cepillo "pin 3", puede obtener un voltaje variable dentro del rango de la fuente de alimentación.



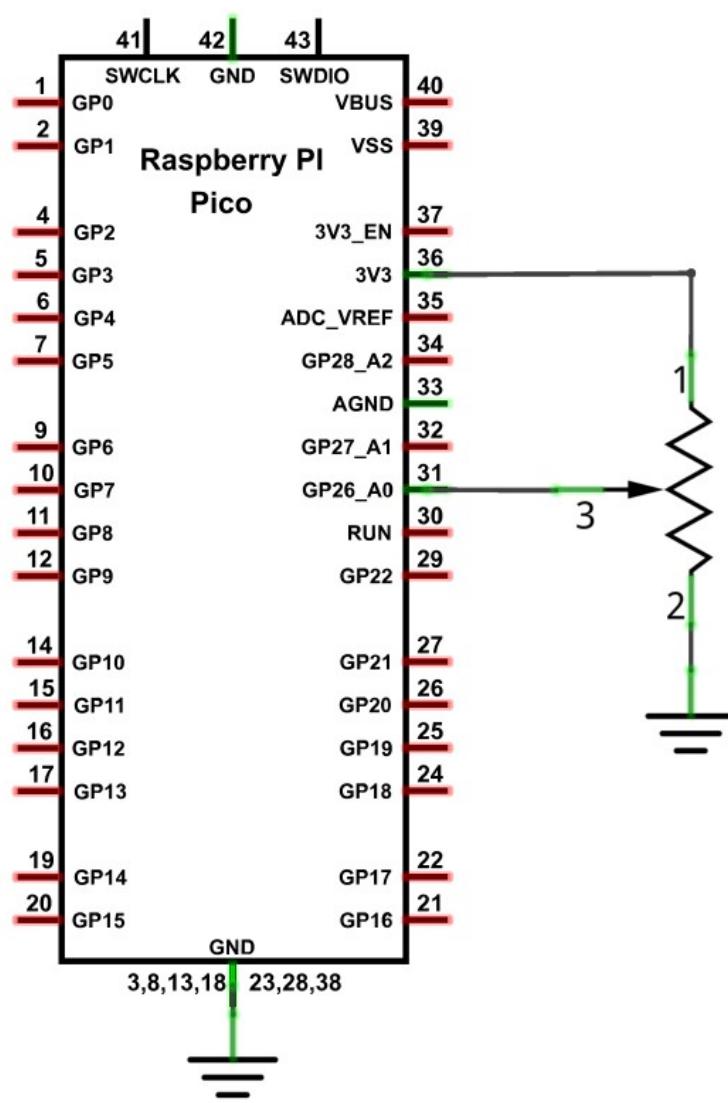
potenciómetro giratorio

Los potenciómetros rotatorios y los potenciómetros lineales tienen la misma función; la única diferencia es que la acción física es un movimiento de rotación en lugar de un movimiento deslizante.

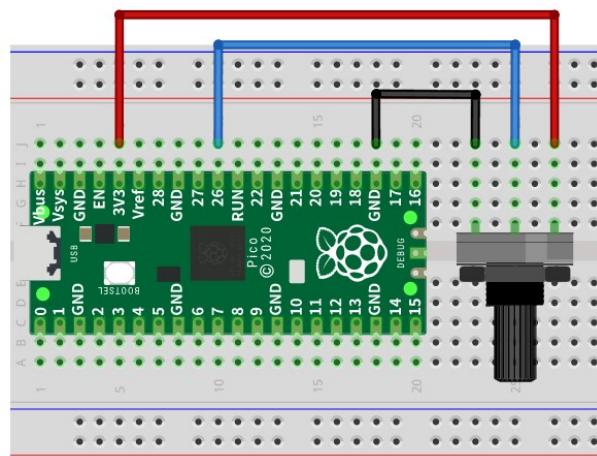


Circuito

Diagrama esquemático



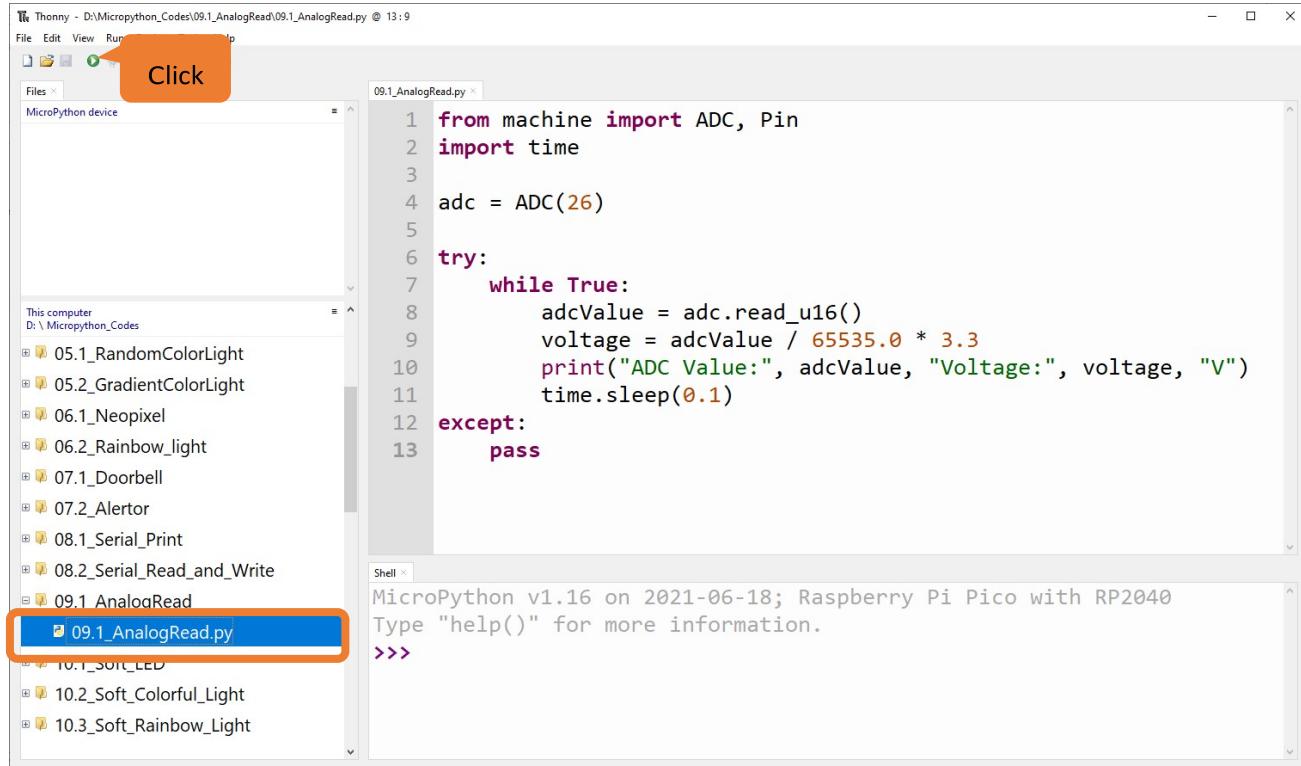
Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



Código

Abra “Thonny”, haga clic en “Esta computadora” → “D:” → “Micropython_Codes” → “08.1_AnalogRead” y luego haga clic en “08.1_AnalogRead.py”.

08.1_AnalogRead



```

from machine import ADC, Pin
import time

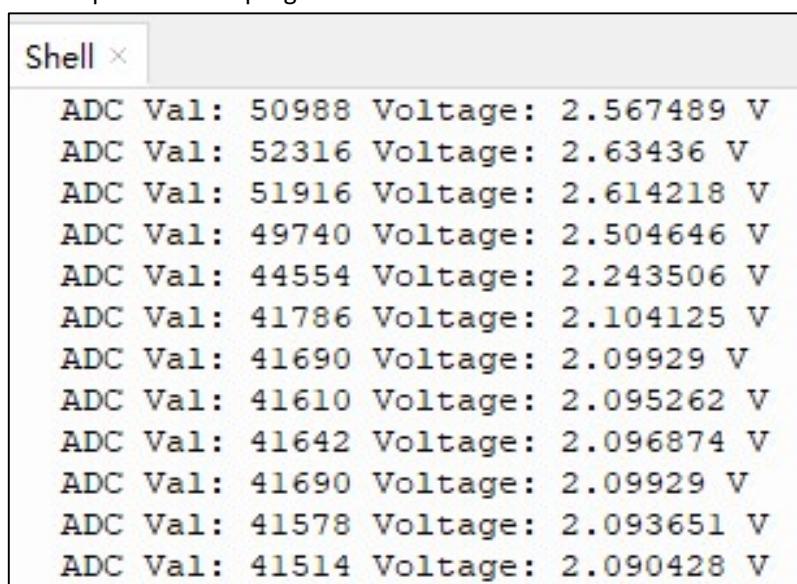
adc = ADC(26)

try:
    while True:
        adcValue = adc.read_u16()
        voltage = adcValue / 65535.0 * 3.3
        print("ADC Value:", adcValue, "Voltage:", voltage, "V")
        time.sleep(0.1)
except:
    pass

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
=>

Haga clic en "Ejecutar script actual" y observe el mensaje impreso en "Shell". Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



```

ADC Val: 50988 Voltage: 2.567489 V
ADC Val: 52316 Voltage: 2.63436 V
ADC Val: 51916 Voltage: 2.614218 V
ADC Val: 49740 Voltage: 2.504646 V
ADC Val: 44554 Voltage: 2.243506 V
ADC Val: 41786 Voltage: 2.104125 V
ADC Val: 41690 Voltage: 2.09929 V
ADC Val: 41610 Voltage: 2.095262 V
ADC Val: 41642 Voltage: 2.096874 V
ADC Val: 41690 Voltage: 2.09929 V
ADC Val: 41578 Voltage: 2.093651 V
ADC Val: 41514 Voltage: 2.090428 V

```

El siguiente es el código:

```

1 desde la importación de la máquina
2 ADC, tiempo de importación de PIN
3 adc = ADC(26)
4 prueba:
5 mientras que es
6 cierto :
7 adcValue = adc.read_u16() voltage = adcValue / 65535.0 * 3.3
8 print ("Valor ADC:", adcValue, "Voltage:", voltage, "V")
9 time.sleep(0.1) excepto: pass
10
11
12
13

```

Importar módulos de Pin, ADC y tiempo.

```

1 desde la importación de la máquina
2 ADC, tiempo de importación de PIN

```

Cree un objeto ADC y conecte GP26, que corresponde al canal ADC0 de Raspberry Pi Pico.

```

4 adc = ADC(26)

```

Lea el valor ADC cada 0,1 segundos. Calcule el voltaje actual según la fórmula $ADCValue = (\text{Voltaje analógico})/3.3*65535$ e imprímalo en "Shell".

```

7 mientras que es cierto :
8 adcValue = adc.read_u16() voltaje = adcValue / 65535.0 * 3.3
9 print ("Valor ADC:", adcValue, "Voltage:", voltaje, "V")
10 time.sleep(0.1)
11

```

Referencia

ADC de clase

Antes de cada uso del módulo ADC, agregue la declaración "**desde la importación de la máquina ADC**" en la parte superior de la archivo python.

máquina.ADC(pir)

canal o **channel_num**) : Crea un objeto ADC asociado con el pin dado.

Por ejemplo pins disponibles son: GP26, GP27, GP28, GP12.

_num : Canal disponible 0, 1, 2, 3, 4.

:

maquina.ADC(0) = maquina.ADC(26) maquina.ADC(1) =

maquina.ADC(27) maquina.ADC(2) = maquina.ADC(28)

ADC.read_16 maquina.ADC(3) = maquina.ADC(29) Se conecta al sensor de temperatura interno.

() : lee el valor ADC actual y lo devuelve, con un rango de 0-65535.

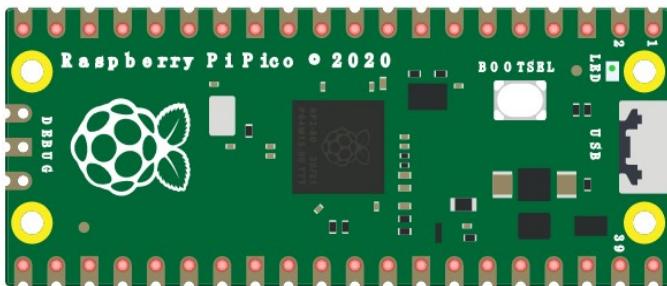
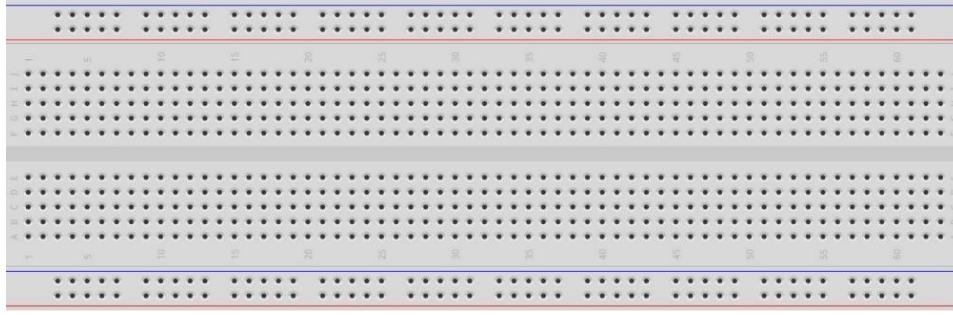
Capítulo 9 Potenciómetro y LED

Hemos aprendido a usar ADC en el capítulo anterior. En este capítulo, combinaremos PWM y ADC para usar un potenciómetro para controlar LED, RGBLED y Neopixel.

Proyecto 9.1 Luz tenue

En este proyecto, haremos una luz suave. Usaremos un módulo ADC para leer los valores ADC de un potenciómetro y asignarlo al ciclo de trabajo del PWM utilizado para controlar el brillo de un LED. Luego puede cambiar el brillo de un LED ajustando el potenciómetro.

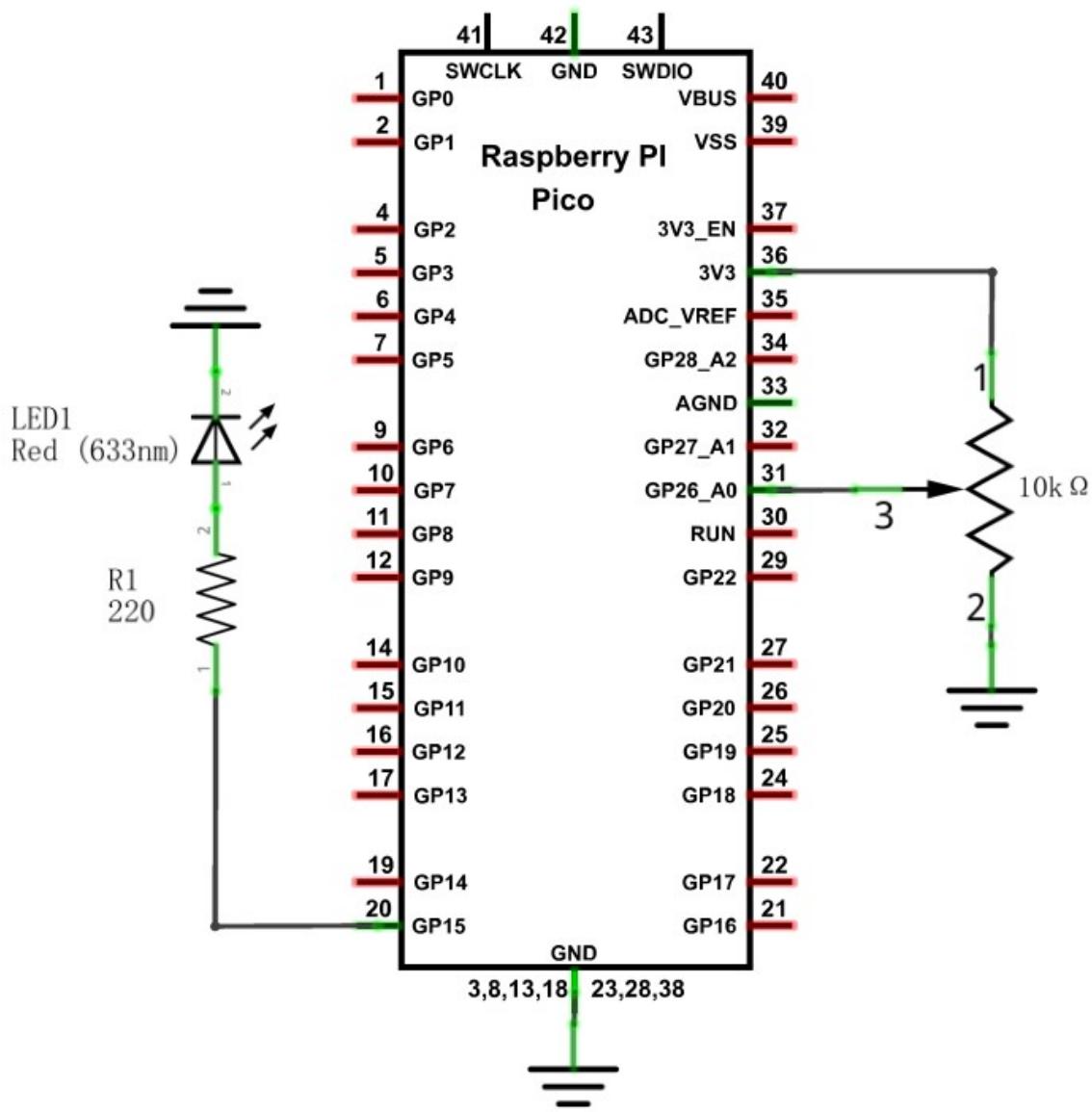
Lista de componentes

Frambuesa Pi Pico x1	Cable USBx1		
			
Protopboard x1			
			
Potenciómetro giratorio x1	Resistencia 220Ω x1	LED x1	Puente
			

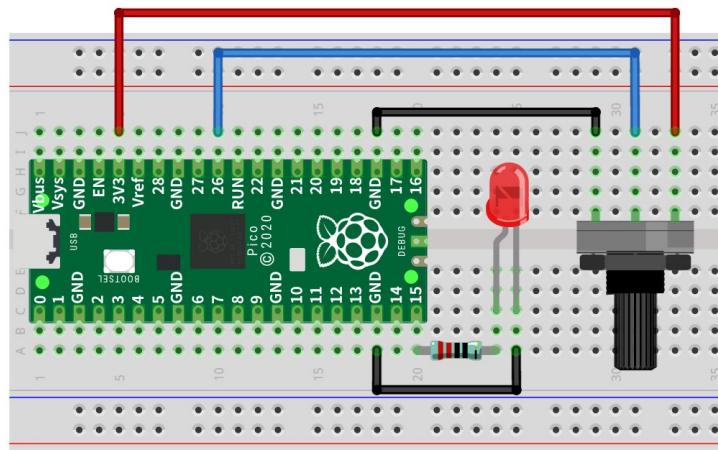
Capítulo 9 Potenciómetro y LED

Circuito

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



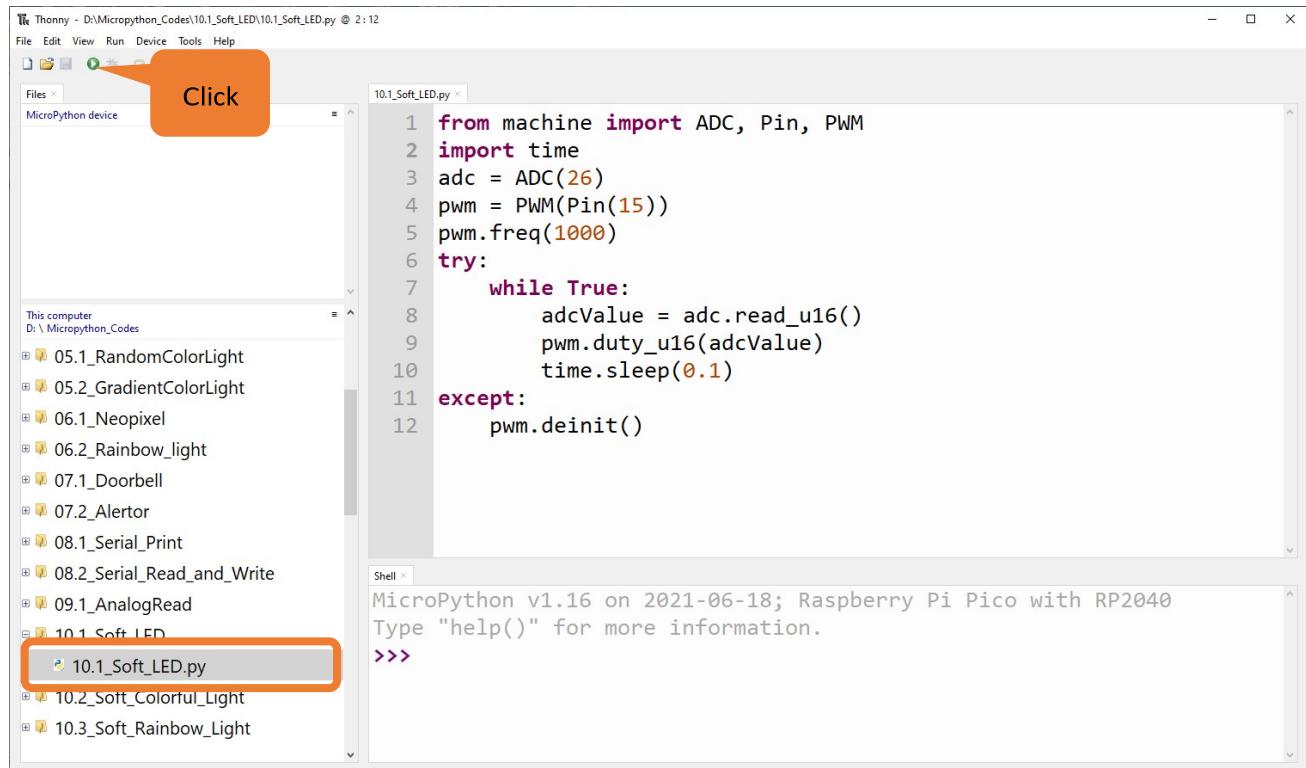
Capítulo 9 Potenciómetro y LED

¿Cualquier duda? support@freenove.com

Código

Abra “Thonny”, haga clic en “ Esta computadora” → “D:” → “Micropython_Codes” → “09.1_Soft_LED” y haga doble clic en “09.1_Soft_LED.py”.

09.1_Soft_LED



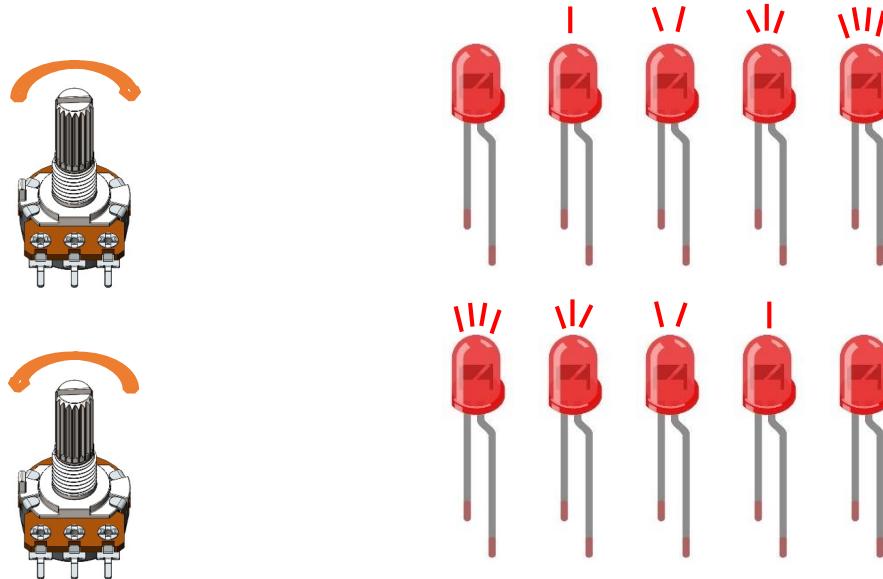
```

from machine import ADC, Pin, PWM
import time
adc = ADC(26)
pwm = PWM(Pin(15))
pwm.freq(1000)
try:
    while True:
        adcValue = adc.read_u16()
        pwm.duty_u16(adcValue)
        time.sleep(0.1)
except:
    pwm.deinit()

```

MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar script actual". Gire el mango del potenciómetro y el brillo del LED cambiará en consecuencia. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



Si usted tiene alguna inquietud, contáctenos a través de: support@freenove.com

Capítulo 9 Potenciómetro y LED El siguiente es el código:

¿Cualquier duda?  support@freenove.com

```
1 desde la máquina importar ADC,
2 Pin, PWM import time
3 adc = ADC(
4    26)
5 pwm = PWM(Pin(15))
6 pwm.freq(1000)
7 try : while True :
8     adcValue = adc.read_u16()
9     pwm.duty_u16(adcValue)
10    time.sleep(0.1)
11 except:
12     pwm.deinit()
```

En el código, lea el valor ADC del potenciómetro y asígnelo al ciclo de trabajo de PWM para controlar el brillo del LED.

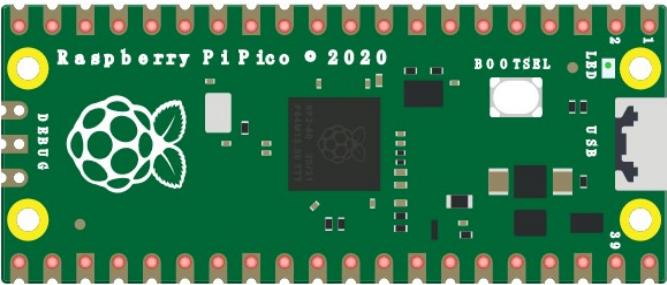
Capítulo 10 Fotorresistor y LED

En este capítulo, aprenderemos cómo usar la fotorresistencia.

Proyecto 10.1 LED de control a través de fotorresistor

Un fotorresistor es muy sensible a la cantidad de luz presente. Podemos aprovechar la característica para hacer una lámpara de noche con la siguiente función: cuando la luz ambiental es menor (ambiente más oscuro) el LED automáticamente se vuelve más brillante para compensar y cuando la luz ambiental es mayor (ambiente más brillante) el LED automáticamente atenuar para compensar.

Lista de componentes

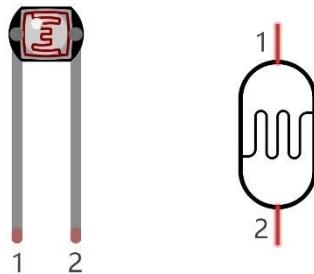
Frambuesa Pi Pico x1		Cable USBx1
Protopboard x1		
Fotorresistencia x1	Resistor 220Ω x1	LED x1
	10KΩ x1	Puente

Capítulo 10 Fotorresistor y LED

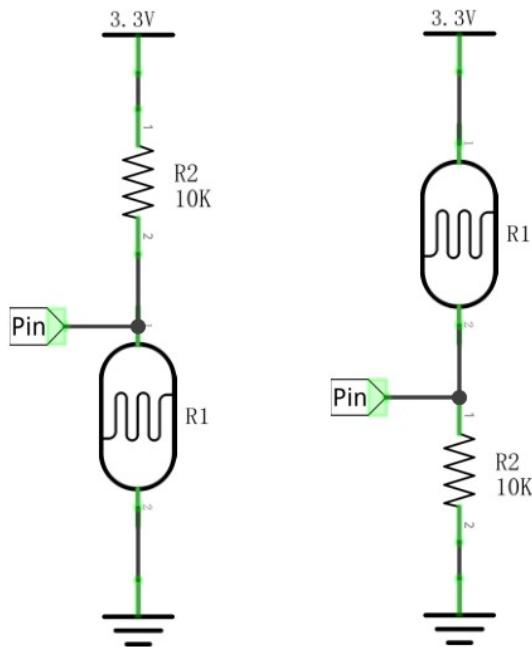
Conocimiento de componentes

fotorresistencia

El fotorresistor es simplemente un resistor sensible a la luz. Es un componente activo que disminuye la resistencia con respecto a recibir luminosidad (luz) en la superficie sensible a la luz del componente. El valor de resistencia del fotoresistor cambiará en proporción a la luz ambiental detectada. Con esta característica, podemos utilizar un Fotorresistor para detectar la intensidad de la luz. El fotorresistor y su símbolo electrónico son los siguientes.



El siguiente circuito se utiliza para detectar el cambio del valor de resistencia de un fotorresistor:

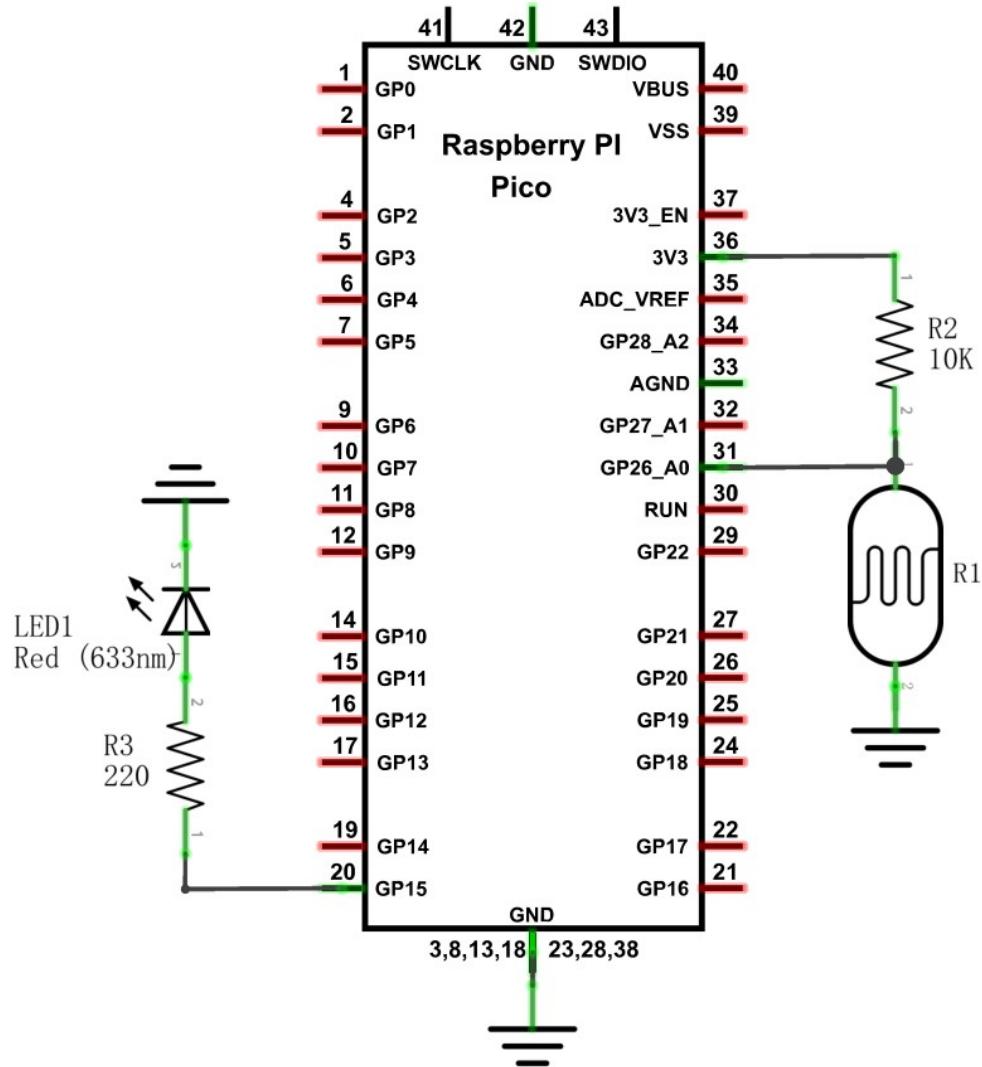


En el circuito anterior, cuando el valor de resistencia de un fotoresistor cambia debido a un cambio en la intensidad de la luz, el voltaje entre el fotorresistor y el resistor R1 también cambiará. Por lo tanto, la intensidad de la luz se puede obtener midiendo este voltaje.

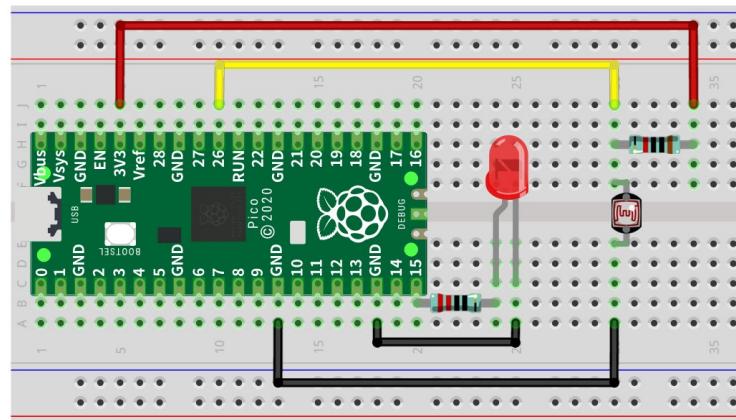
Circuito

El circuito de este proyecto es similar a SoftLight. La única diferencia es que la señal de entrada cambia de un potenciómetro a una combinación de una fotorresistencia y una resistencia.

Diagrama esquemático



Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



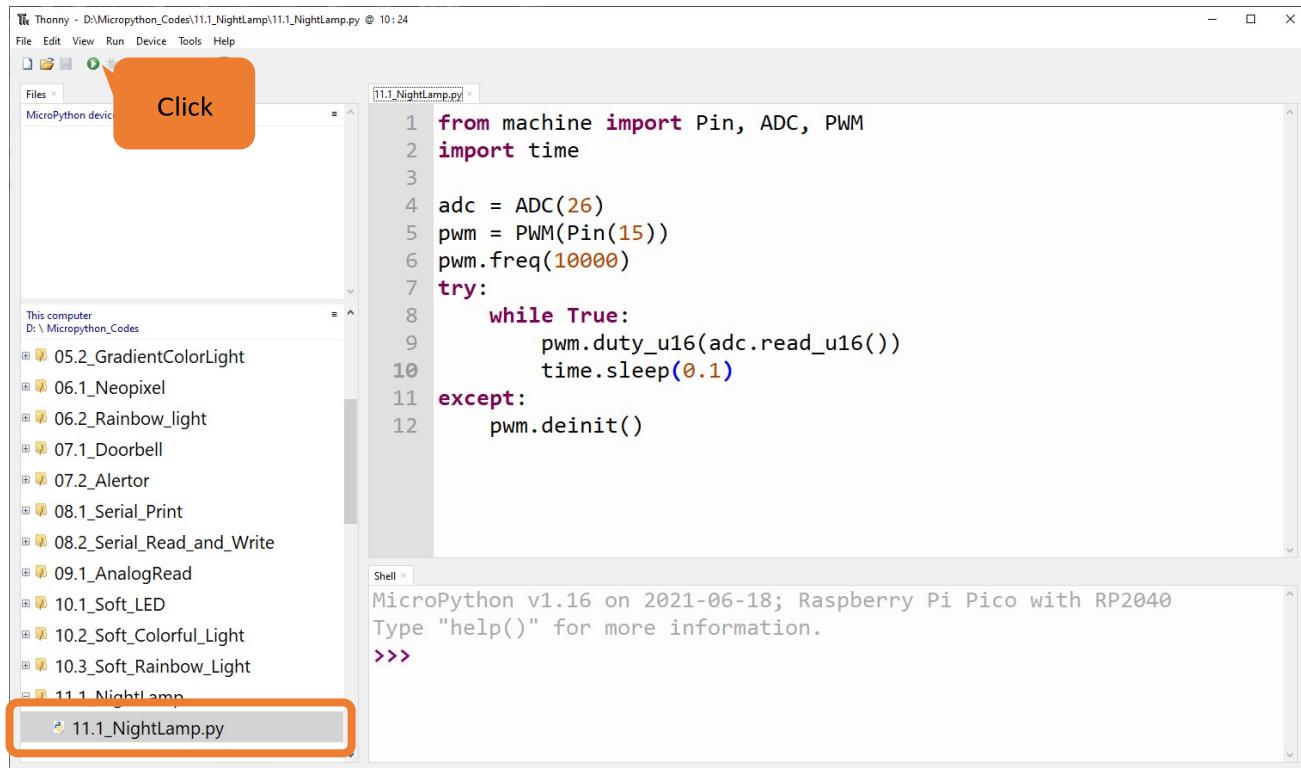
Capítulo 10 Fotorresistor y LED

Código

Los códigos de este proyecto son lógicamente los mismos que los del proyecto [Soft Light](#).

Abra “Thonny”, haga clic en “Esta computadora” → “D:” → “Micropython_Codes” → “10.1_Photoresistor” y haga doble clic en “10.1_Photoresistor.py”.

10.1_Photoresistor



```

from machine import Pin, ADC, PWM
import time

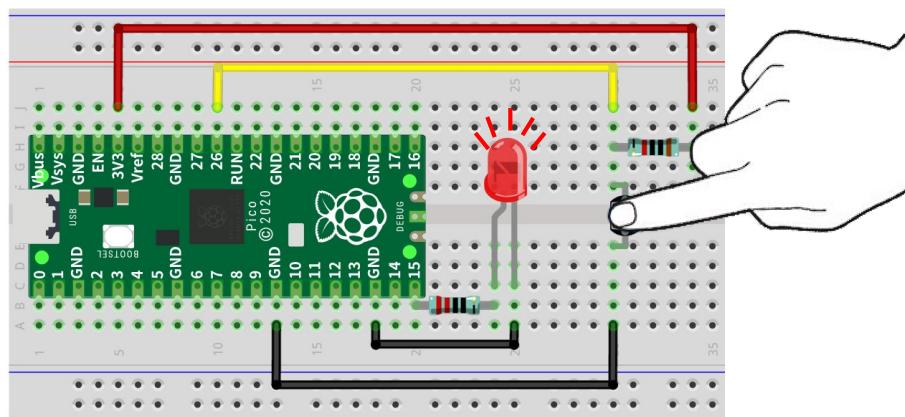
adc = ADC(26)
pwm = PWM(Pin(15))
pwm.freq(10000)
try:
    while True:
        pwm.duty_u16(adc.read_u16())
        time.sleep(0.1)
except:
    pwm.deinit()

```

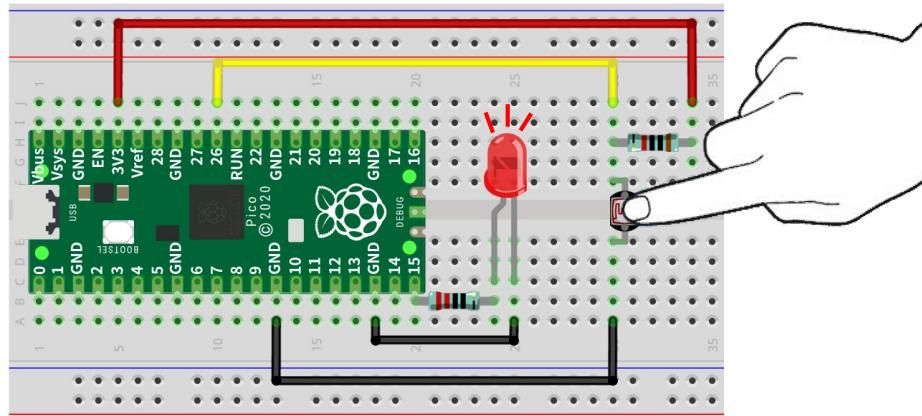
MicroPython v1.16 on 2021-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>

Haga clic en "Ejecutar script actual". Cubra el fotorresistor con sus manos o ilumínelo con luces, el brillo de LEDs will change.

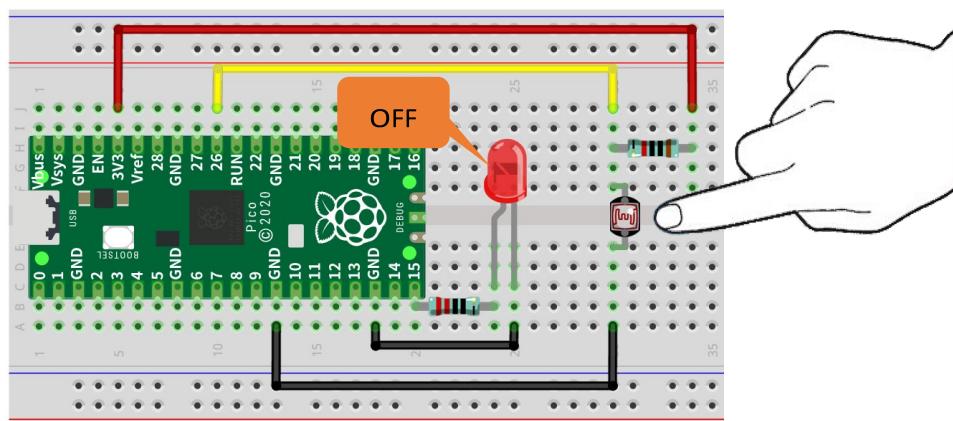
Fully cover the photoresistor :



Cubre la mitad de la fotorresistencia :



No cubra la fotorresistencia:



El siguiente es el código del programa:

```

1 desde la importación de la máquina Pin,
2 ADC, tiempo de importación de PWM
3 adc = ADC( 26 )
4 pwm =
5 PWM(Pin(15))
6 pwm.freq(10000)
7 prueba : while
8 True :
9     pwm.duty
10    _u16(adc.read_u16())
11    time.sleep(0.1) excepto :
12        pwm.deinit ()
```

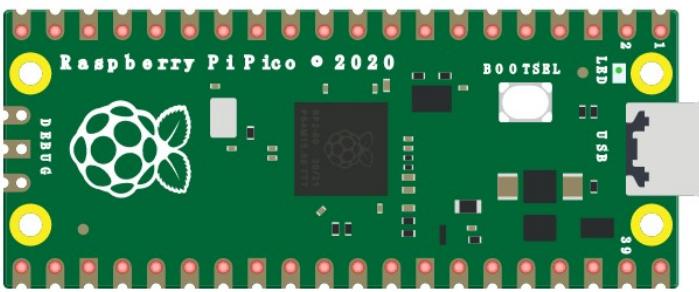
Capítulo 11 Termistor

En este capítulo, aprenderemos sobre los termistores, que son otro tipo de resistencia.

Proyecto 11.1 Termómetro

Un termistor es un tipo de resistencia cuyo valor de resistencia depende de la temperatura y los cambios de temperatura. Por tanto, podemos aprovechar esta característica para fabricar un Termómetro.

Lista de componentes

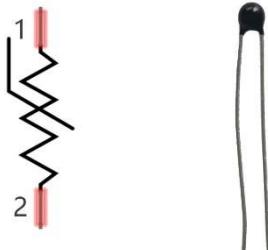
Frambuesa Pi Pico x1		Cable USBx1
Protopboard x1		
Termistor x1		Resistencia 10kΩ x1



Conocimiento de componentes

termistor

Un termistor es una resistencia sensible a la temperatura. Cuando detecta un cambio de temperatura, la resistencia del termistor cambiará. Podemos aprovechar esta característica utilizando un termistor para detectar la intensidad de la temperatura. A continuación se muestra un termistor y su símbolo electrónico.



The relationship between resistance value and temperature of a thermistor is:

$$R_t = R * \text{EXP} \left[B * \left(\frac{1}{T_2} - \frac{1}{T_1} \right) \right]$$

Where:

R_t es la resistencia del termistor a la temperatura T₂;

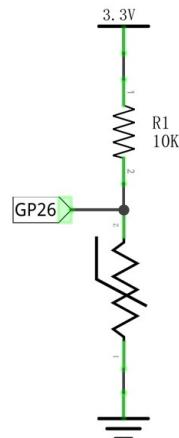
R es la resistencia nominal del termistor bajo la temperatura T₁;

EXP[n] es la n-ésima potencia de e; **B** es para el índice térmico;

T₁, T₂ es la temperatura Kelvin (temperatura absoluta). Temperatura Kelvin=273,15 + temperatura Celsius.

Para los parámetros del termistor, usamos: B=3950, R=10kΩ, T₁=25°C.

El método de conexión del circuito del termistor es similar al de la fotorresistencia, como se muestra a continuación:



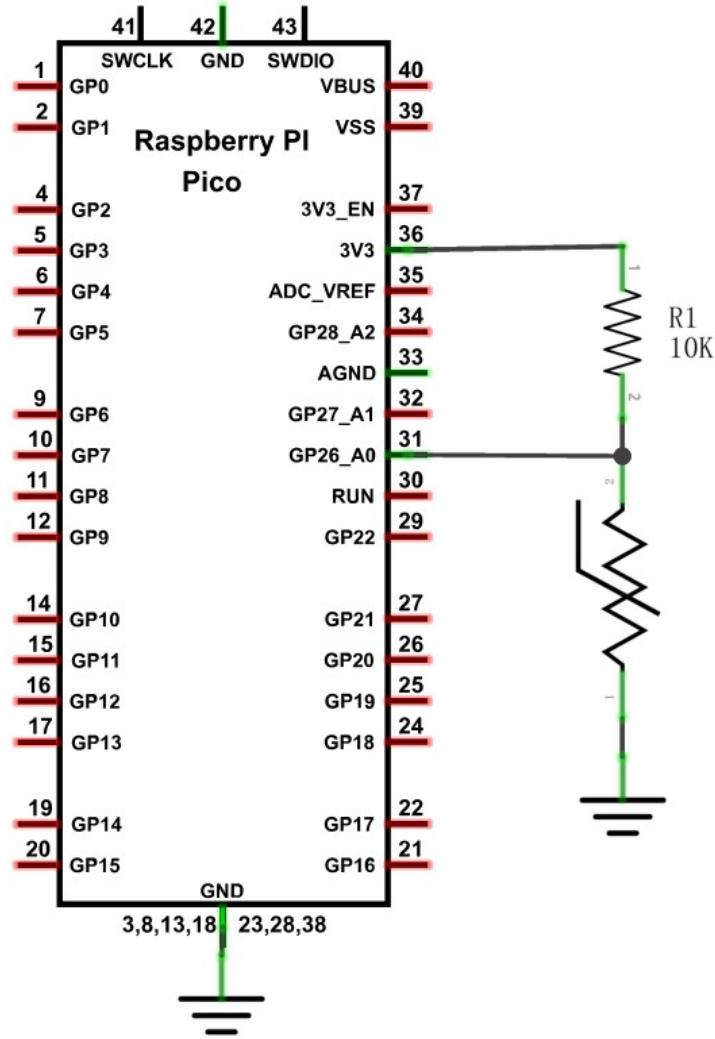
Podemos usar el valor medido por el convertidor ADC para obtener el valor de resistencia del termistor, y luego podemos usar la fórmula para obtener el valor de temperatura. Por lo tanto, la fórmula de temperatura se puede derivar como:

$$T_2 = 1 / \left(\frac{1}{T_1} + \ln \left(\frac{R_t}{R} \right) / B \right)$$

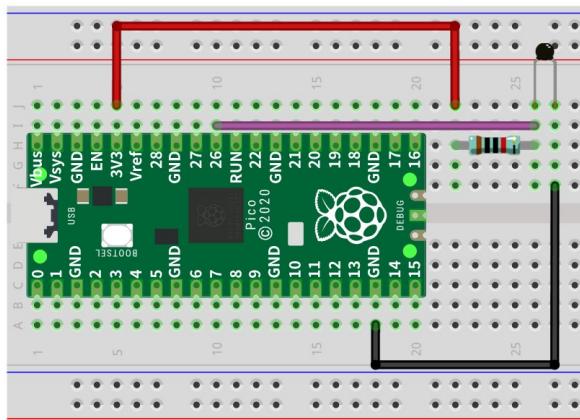
Circuito

El circuito de este proyecto es similar al del capítulo anterior. La única diferencia es que el fotorresistor se reemplaza por un termistor.

Diagrama esquemático



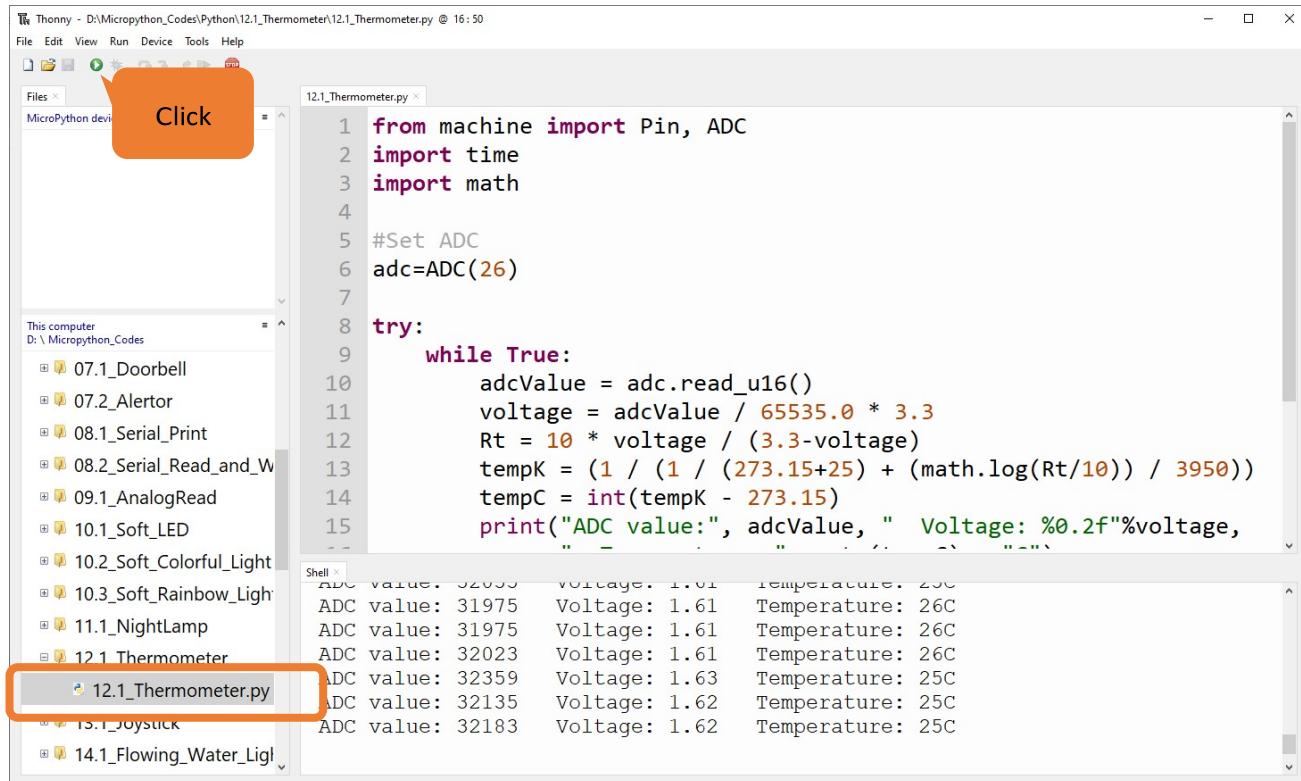
Conexión de hardware. Si necesita ayuda, no dude en contactarnos a través de: support@freenove.com



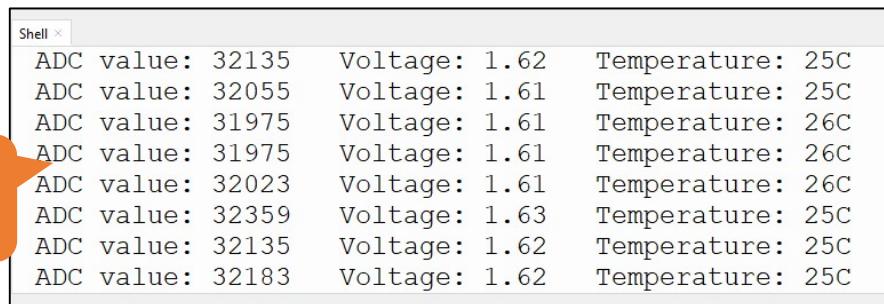
Código

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "11.1_Thermometer" y haga doble clic en "11.1_Thermometer.py".

11.1 Thermometer



Haga clic en "Ejecutar secuencia de comandos actual" y "Shell" mostrará constantemente el valor actual de ADC, el valor de voltaje y el valor de temperatura. Intente "pellizcar" el termistor (sin tocar los cables) con su dedo índice y pulgar por un breve tiempo, debería ver que el valor de la temperatura aumenta. Presione Ctrl+C o haga clic en "Detener/Reiniciar backend" para salir del programa.



Si usted tiene alguna inquietud, contáctenos a través de: support@freenove.com

El siguiente es el código:

```

1 importación de
2 máquina , tiempo de
3 importación de ADC
4 Matemáticas de
5 importación
6 #Establecer
7 ADC adc= ADC(
8     26)
9     prueba:
10    mientras
11        que es
12        cierto:
13            adcValue = adc.read_u16() voltaje = adcValue / 65535.0 *
14            3.3 Rt = 10 * voltaje / (3.3-voltaje) tempK = (1 / (1 /
15            (273.15+25) + (math.log(Rt/10)) / 3950)) tempC = int(tempK
dieciséis
16            - 273.15) print ("Valor ADC:", adcValue, "Voltaje:
17            %.2f"%voltaje,
18            " Temperatura : " + str
19            (tempC) + " C ") time.sleep(1) excepto :
pass

```

Se llama a la función `read_u16()` para leer el valor de ADC0.

```
10    adcValue = adc.read_u16()
```

Convierta el valor leído de ADC0 para obtener el valor de voltaje del termistor actual.

```
11    voltaje = adcValue / 65535.0 * 3.3
```

La resistencia actual del termistor (Rt) se calcula mediante la ley de Ohm.

```
12    Rt = 10 * voltaje / (3.3-voltaje)
```

Según la fórmula: $T_2 = 1/\left(\frac{1}{T_1} + \ln\left(\frac{R_t}{R}\right)/B\right)$, donde $T_1 = 25^\circ\text{C}$, $R = 10\text{K}\Omega$, $B = 3950$ y el Rt calculado

en el paso anterior, sustituya la fórmula para calcular `tempK(T2)`. Obtenga el valor de la unidad de temperatura K.

```
13    tempK = (1 / (1 / (273.15+25) + (math.log(Rt/10)) / 3950)))
```

Finalmente, `tempK` (unidad: K) se convierte a `tempC` (unidad: $^\circ\text{C}$). También puede convertir a Fahrenheit según sus necesidades.

```
14    tempC = int(tempK - 273.15)
```

[Capítulo 12 Modos de trabajo WiFi \(solo para Pico W\)](#)

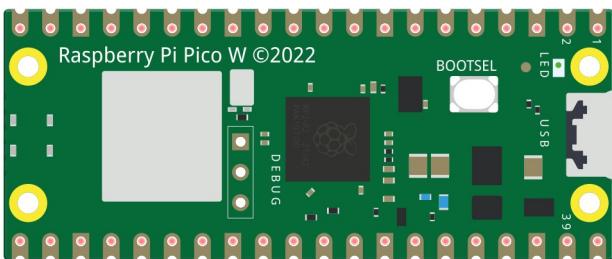
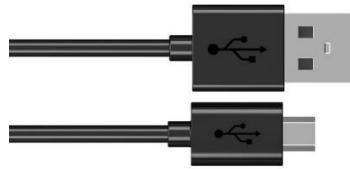
Capítulo 12 Modos de trabajo WiFi (solo para Pico W)

La mayor diferencia entre raspberry pi ico y raspberry pi pico W es que raspberry pi Pico W está equipado con un módulo de función WiFi. Al comienzo de este capítulo, aprenderemos sobre la función WiFi de Pico W de Raspberry Pi.

Si tiene Pico en la mano, cámbielo a Pico W antes de continuar con el aprendizaje.

Proyecto 12.1 Modo estación

Lista de componentes

Frambuesa Pi Pico W x1	Cable micro USB x1
	

Conocimiento de componentes

Inalámbrica

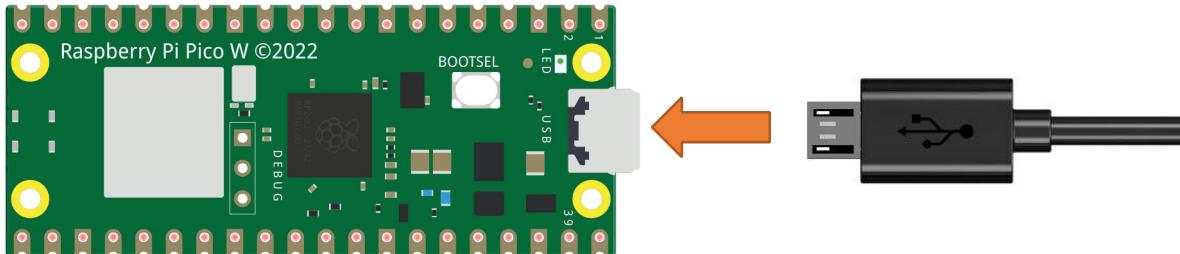
Pico W tiene una interfaz inalámbrica integrada de 2,4 GHz que utiliza un Infineon CYW43439. La antena es una antena integrada con licencia de ABRACON (anteriormente ProAnt). La interfaz inalámbrica está conectada a través de SPI al RP2040. [modo estación](#)

Cuando PICO W selecciona el modo Estación, actúa como un cliente WiFi. Puede conectarse a la red del enrutador y comunicarse con otros dispositivos en el enrutador a través de una conexión WiFi. Como se muestra a continuación, la PC está conectada al enrutador, y si PICO W quiere comunicarse con la PC, debe estar conectado al enrutador.



Circuito

Connect Pico W to the computer using the USB cable.



Código

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes** " al disco (D) de antemano con la ruta de " **D:/Micropython_Codes** ".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "12.1_Station_mode" y haga doble clic en "12.1_Station_mode.py".

Capítulo 12 Modos de trabajo WiFi (solo para Pico W)

12.1_Station_mode

```

Thonny - C:\Users\DESKTOP-LIN\Desktop\clear\Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi
File Edit View Run Tools Help
Files x
This computer
C: \ Users \ DESKTOP-LIN \
Desktop \ clear \
Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi_Pico \ Python \
Python_Codes \
29.1_Station_mode
29.1_Station_mode.py x
29.1_Station_mode.py
1 import time
2 import network
3
4 ssidRouter      = '*****' #Enter the router name
5 passwordRouter = '*****' #Enter the router password
6
7 def STA_Setup(ssidRouter,passwordRouter):
8     print("Setup start")
9     sta_if = network.WLAN(network.STA_IF)
10    if not sta_if.isconnected():
11        print('connecting to',ssidRouter)
12        sta_if.active(True)
13        sta_if.connect(ssidRouter,passwordRouter)
14        while not sta_if.isconnected():
15            pass
16        print('Connected, IP address:', sta_if.ifconfig())
17        print("Setup End")
18
19 try:
20     STA_Setup(ssidRouter,passwordRouter)
21 except:
22     sta_if.disconnect()
23
Shell x
MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>>
MicroPython (Raspberry Pi Pico)

```

Debido a que los nombres y contraseñas de los enruteadores en varios lugares son diferentes, antes de que se ejecute el Código, los usuarios deben ingresar el nombre y la contraseña correctos del enruteador en el cuadro como se muestra en la ilustración anterior.

Después de asegurarse de que el nombre y la contraseña del enruteador se ingresaron correctamente, compile y cargue los códigos en PICO W, espere a que PICO W se conecte a su enruteador e imprima la dirección IP asignada por el enruteador a PICO W en "Shell".

```

Shell x
MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
Setup start
Connected, IP address: ('192.168.1.26', '255.255.255.0', '192.168.1.1', '192.168.1.1')
Setup End
>>>

```

El siguiente es el código del programa:

1	tiempo de
2	importación
3	red de
4	importación
5	ssidRouter = '*****' #Ingrese el nombre del enruteador passwordRouter = '*****' #Ingrese la contraseña del enruteador

```

7   definitivamente STA_Setup (
8     ssidRouter, passwordRouter ) :
9     imprimir ( "Iniciar configuración"
10    ) sta_if = red.WLAN(red.STA_IF) si
11    no sta_if.isconnected():
12      print ( 'conectando a ',ssidRouter)
13      sta_if.active( True )
14      sta_if.connect(ssidRouter,passwordRouter) while
15    no sta_if.isconnected():
16      pasar           print ( 'Conectado,
diecisési
s
17      dirección IP: ', sta_if.ifconfig() ) print ( "Fin
de la configuración" )

18
19      p
20      r
21      u
22      e
23      b
24      a
25      :
26      STA_Setup( ssidRouter, passwordRouter )
27      excepto :
28        sta_si.desconectar ()

```

Importar módulo de red.

2	importar red
---	--------------

Ingresar el nombre y la contraseña correctos del enrutador.

4	constante char *ssid_Router = "*****" ; //Ingresar el nombre del 5 enrutador constante char *contraseña_Router = "*****" ; //Ingresar la contraseña del enrutador
---	---

Establezca PICO W en modo Estación.

9	sta_if = red.WLAN (red.STA_IF)
---	----------------------------------

Active el modo Estación de Pico W, inicie una solicitud de conexión al enrutador e ingrese la contraseña para conectarse.

12	sta_if.activo (Verdadero) _
13	sta_if.connect (ssidRouter ,passwordRouter)

Espere a que PICO W se conecte al enrutador hasta que se conecten entre sí con éxito.

14	tiempo no sta_if.isconnected ():
15	pasar

Imprime la dirección IP asignada a PICO W en "Shell".

diecisésis	Imprimir ('Conectado, dirección IP: ', sta_if.ifconfig())
------------	--

Referencia

red de clases	
---------------	--

Antes de cada uso:

WLAN (interfaz) : red , agregue la declaración " **importar red** " en la parte superior del archivo python.

red.STA_IF red. : Establecer en modo Wi-Fi.

active(is_active) : Cliente, conectándose a otros puntos de acceso WiFi.

es_scan(ssid) : Puntos de acceso, permitiendo que otros clientes WiFi se conecten.

bssid scan en STA : Con parámetros, es para verificar si activar la interfaz de red; Sin consultar el estado actual de la interfaz de red.

, canal, RSSI, modo de autenticación, oculto : busca redes inalámbricas disponibles en las canas (solo interfaz), devuelve una lista de tuplas de información sobre el punto de acceso WiFi.

dirección de hardware del punto de acceso, devuelta en forma binaria como un objeto de byte.

puedes usar

Capítulo 12 Modos de trabajo WiFi (solo para Pico W)

ubinascii.hexlify () para convertirlo a formato ASCII.

authmode : tipo de acceso

AUTORIZACIÓN_ABIERTO = 0

AUTH_WEP = 1

AUTH_WPA_PSK = 2

AUTH_WPA2_PSK = 3

AUTH_WPA_WPA2_PSK = 4

AUTH_MAX = 6

Oculto: si buscar puntos de acceso ocultos

Falso : solo busca puntos de acceso visibles

Verdadero : exploración de todos los puntos de acceso, incluidos los ocultos.

isconnected() : Compruebe si PICO W está conectado a AP en modo Estación. En modo STA, devuelve True si está conectado a un punto de acceso WiFi y tiene una dirección IP válida; De lo contrario, devuelve False.

connect (ssid, contraseña) : Conexión a la red inalámbrica.

ssid : WiFi name **contraseña**

: WiFi password

desconectar() : desconectarse de la red inalámbrica actualmente conectada.

Proyecto 12.2 Modo AP

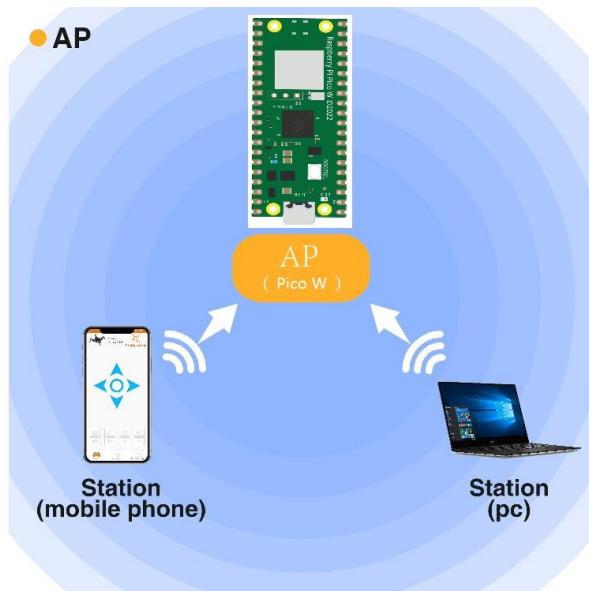
Lista de componentes y circuito

La lista de componentes y el circuito son los mismos que en la Sección 12.1.

Conocimiento de componentes

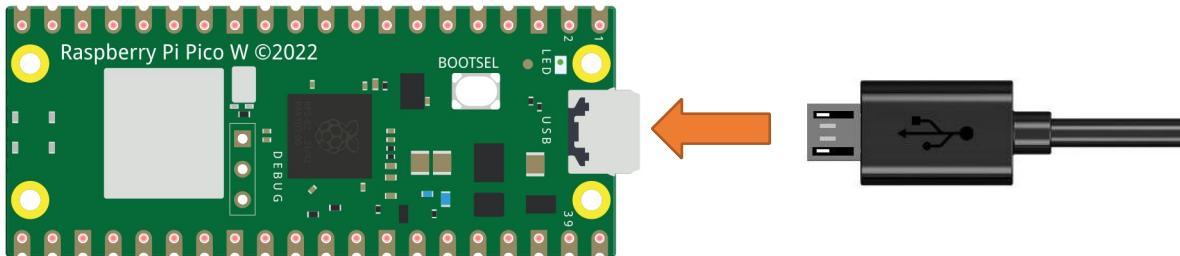
Modo AP

Cuando PICO W selecciona el modo AP, crea una red de punto de acceso que está separada de Internet y espera a que se conecten otros dispositivos WiFi. Como se muestra en la figura a continuación, PICO W se utiliza como punto de acceso. Si un teléfono móvil o PC quiere comunicarse con PICO W, debe estar conectado al punto de acceso de PICO W. Solo después de establecer una conexión con PICO W pueden comunicarse.



Circuito

Connect Pico W to the computer using the USB cable.



Código

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes** " al disco (D) de antemano con la ruta de " **D:/Micropython_Codes** ".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "12.2_AP_mode". y haga doble clic en "12.2_AP_mode.py".

12.2_AP_mode

```

import network
ssidAP      = 'WiFi_Name' #Enter the router name
passwordAP   = '12345678' #Enter the router password
local_IP     = '192.168.1.10'
gateway      = '192.168.1.1'
subnet       = '255.255.255.0'
dns          = '8.8.8.8'

ap_if = network.WLAN(network.AP_IF)

def AP_Setup(ssidAP,passwordAP):
    ap_if.disconnect()
    ap_if.ifconfig([local_IP,gateway,subnet,dns])
    print("Setting soft-AP ... ")
    #ap_if.config(essid=ssidAP,authmode=network.AUTH_WPA_WPA2_PSK, password=passwordAP)
    ap_if.config(essid=ssidAP, password=passwordAP)
    ap_if.active(True)
    print('Success, IP address:', ap_if.ifconfig())
    print("Setup End\n")

```

Set a name and a password for Pico W AP Mode.

Antes de que se ejecute el código, puede realizar cualquier cambio en el nombre y la contraseña de AP para PICO W en el cuadro como se muestra en la ilustración anterior. Por supuesto, puede dejarlo solo por defecto.

Haga clic en "Ejecutar script actual" [abre la función AP de PICO W e imprima la información del punto de acceso.](#)

```

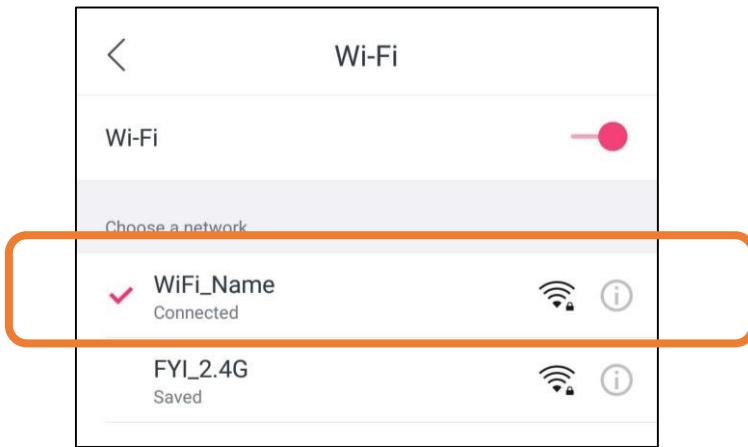
MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Setting soft-AP ...
Success, IP address: ('192.168.4.1', '255.255.255.0', '192.168.4.1
', '8.8.8.8')
Setup End

>>>

```

Active la función de escaneo WiFi de su teléfono y podrá ver el ssid_AP en PICO W, que se denomina "WiFi_Name" en este Código. Puede ingresar la contraseña "12345678" para conectarlo o cambiar su nombre de AP y contraseña modificando el Código.



El siguiente es el código del programa:

```

1 importar red
2 ssidAP = 'WiFi_Name' #Ingrese el nombre del
3 enrutador passwordAP = '12345678' #Ingrese la
4 contraseña del enrutador
5 IP_local = '192.168.1.10'
6 puerta de enlace =
7 '192.168.1.1' subred =
8 '255.255.255.0' dns =
9 '8.8.8.8'
10 ap_if = red.WLAN(red.AP_IF)
11 definitivamente AP_Setup (ssidAP,
12 passwordAP):
13 ap_if.ifconfig([local_IP, gateway, subnet, dns])
14 print ( "Configuración de soft-AP..." )
15 ap_if.config(essid=ssidAP,
16 contraseña=passwordAP) ap_if.active( True )
17     print ( 'Éxito, dirección IP:' , ap_if.ifconfig() ) print
18 ( "Fin de la configuración\n" )
19 p
20 r
21 u
22 e
23 b
24 a
:
```

```
AP_Setup(ssidAP, contraseñaAP)
excepto :
    ap_if.desconectar()
```

Importar módulo de red.

```
1 importar red
```

Ingresar el nombre de AP y la contraseña correctos.

```
3 ssidAP = 'WiFi_Name' #Ingresar el nombre del enrutador
4 passwordAP = '12345678' #Ingresar la contraseña del enrutador
```

Configurar PICO W en modo AP.

```
11 ap_if = red.WLAN(red.AP_IF)
```

Configurar la dirección IP, la puerta de enlace y la máscara de subred para PICO W.

```
14 ap_if.ifconfig([local_IP, puerta de enlace, subred, dns])
```

Encienda un AP en PICO W, cuyo nombre se establece mediante ssid_AP y la contraseña se establece mediante password_AP.

```
dieciséis ap_if.config(essid=ssidAP, contraseña=contraseñaAP)
17 ap_if.active(Verdadero)
```

Si el programa se ejecuta de manera anormal, se llamará a la función de desconexión de AP.

```
14 ap_if.desconectar()
```

Referencia

red de clases

Antes de cada uso:

WLAN (interfaz red, agregue la declaración " importar red " en la parte superior del archivo python.

network.S1 : Establecer en modo Wi-Fi.

network.AI : Cliente, conectándose a otros puntos de acceso WiFi

active(is_acceso) Puntos de acceso, lo que permite que otros clientes WiFi se conecten

parámetros: Con parámetros, es para verificar si activar la interfaz de red; Sin consultar el estado actual de la interfaz de red

connect(ssid, password): En modo AP devuelve True si está conectado a la estación; de lo contrario, devuelve False.

WiFi en **contraseña** : Conexión a la red inalámbrica

config(ssid, nombre)

nombre del WiFi **password channel** : para obtener la dirección MAC del punto de acceso o para configurar

ssid : WiFi canal canal WiFi y el punto de acceso.

ifconfig([(ip, nombre de la cuenta)

servidor_DNS) ip, channel subnet, gateway, dns]): Sin parámetros, devuelve una tupla de 4 (ip, subnet_mask, dirección IP, ; Con parámetros, configura IP estática.

puerta de enlace

máscara_subred : entrada de máscara

Desconexión de red subred

servidor_DNS() : El servidor DNS está conectado desde la red inalámbrica

Estado D () actualmente conectada

Vuelve el estado actual de la conexión inalámbrica

Proyecto 12.3 Modo AP+Estación

Lista de componentes y circuito

La lista de componentes y el circuito son los mismos que en la Sección 12.1.

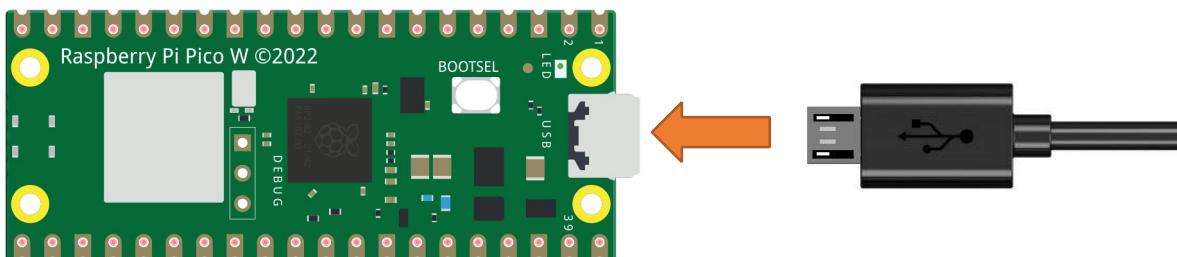
Conocimiento de componentes

Modo AP+Estación

Además del modo AP y el modo Estación, PICO W también puede usar el modo AP y el modo Estación al mismo tiempo. Este modo contiene las funciones de los dos modos anteriores. Active el modo de estación de PICO W, conéctelo a la red del enrutador y podrá comunicarse con Internet a través del enrutador. Al mismo tiempo, encienda su modo AP para crear una red de punto de acceso. Otros dispositivos WiFi pueden optar por conectarse a la red del enrutador o a la red del punto de acceso para comunicarse con PICO W.

Circuito

Connect Pico W to the computer using the USB cable.



Código

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes** " al disco (D) de antemano con la ruta de " **D:/Micropython_Codes** ".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "12.3_AP+STA_mode" y haga doble clic en "12.3_AP+STA_mode.py".

12.3_AP+STA_mode

The screenshot shows the Thonny IDE interface with the file "29.3_AP+STA_mode.py" open. The code is as follows:

```

1 import network
2
3 ssidRouter      = '*****'      #Enter the router name
4 passwordRouter = '*****' #Enter the router password
5
6 ssidAP         = 'Pico W'      #Enter the AP name
7 passwordAP     = '12345678'    #Enter the AP password
8
9 local_IP       = '192.168.4.150'
10 gateway        = '192.168.4.1'
11 subnet         = '255.255.255.0'
12 dns            = '8.8.8.8'
13
14 sta_if = network.WLAN(network.STA_IF)
15 ap_if = network.WLAN(network.AP_IF)
16

```

A callout box highlights the configuration parameters (ssidRouter, passwordRouter, ssidAP, passwordAP) and contains the following note:

Please enter the correct names and passwords of Router and AP.

Es análogo al Proyecto 12.1 y al Proyecto 12.2. Antes de ejecutar el Código, debe modificar ssidRouter, passwordRouter, ssidAP y passwordAP que se muestran en el cuadro de la ilustración anterior.

Después de asegurarse de que el código se modificó correctamente, haga clic en "Ejecutar secuencia de comandos actual" y el "Shell" se mostrará de la siguiente manera:

```
Shell < 
>>>

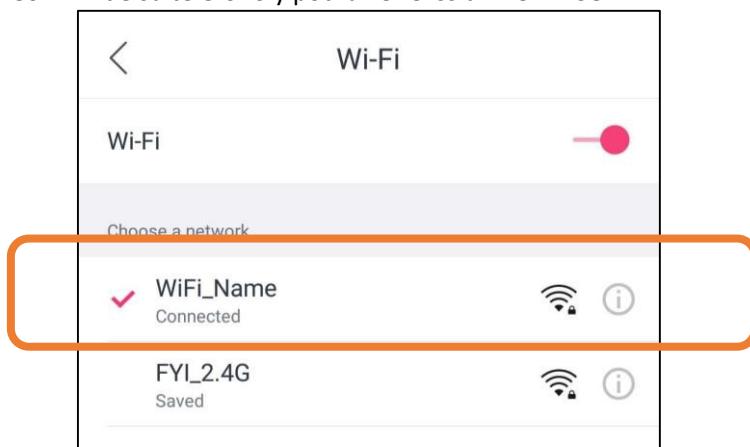
MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Setting soft-AP ...
Success, IP address: ('192.168.4.150', '192.168.4.1', '255.255.255.0', '8.8.8.8')
Setup End

Setting soft-STA ...
Connected, IP address: ('192.168.1.26', '255.255.255.0', '192.168.1.1', '8.8.8.8')
Setup End

>>>
```

Active la función de escaneo WiFi de su teléfono y podrá ver el ssidAP en PICO W.



El siguiente es el código del programa:

```
1 importar red
2 ssidRouter = '*****' #Ingrese el nombre del
3 enrutador passwordRouter = '*****' #Ingrese la
4 contraseña del enrutador
5 ssidAP = 'WiFi_Name'#Ingrese el nombre del AP
6 passwordAP = '12345678' #Ingrese la
7 contraseña del AP
8 IP_local = '192.168.4.150'
9 puerta de enlace =
10 '192.168.4.1' subred =
11 '255.255.255.0' dns =
12 '8.8.8.8'
13 sta_if = red.WLAN(red.STA_IF)
14 ap_if = red.WLAN(red.AP_IF)
15     definitivamente STA_Setup
diecisíe
16 (ssidRouter, contraseñaRouter):
17     print ("Configuración de
soft-STA...") si no
18     sta_if.isconnected():
19         print ('conectando a', ssidRouter)
20     sta_if.active (True) sta_if.connect
21 (ssidRouter,passwordRouter) while no
22     sta_if.isconnected():
23         pasar print ('Conectado, dirección
IP:', sta_if.ifconfig()) print ("Fin de la
24 configuración")
25     definitivamente AP_Setup
26 (ssidAP,passwordAP):
27     ap_if.ifconfig([local_IP,gateway,subnet,dns])
28 ) print ("Configuración de soft-AP...")
29
30
31 ap_if.config(essid=ssidAP,
32 contraseña=contraseñaAP) ap_if.active( True )
33 print ('Éxito, dirección IP:',
34 ap_if.ifconfig()) print ("Fin de
35 configuración\n")
36
37 p
38 r
39 u
40 e
41 b
```

```
a
:
AP_Setup(ssidAP, passwordAP)
STA_Setup(ssidRouter, passwordRouter)
excepto :
sta_if.desconectar()
ap_if.desconectar()
```

Capítulo 13 TCP/IP (solo para Pico W)

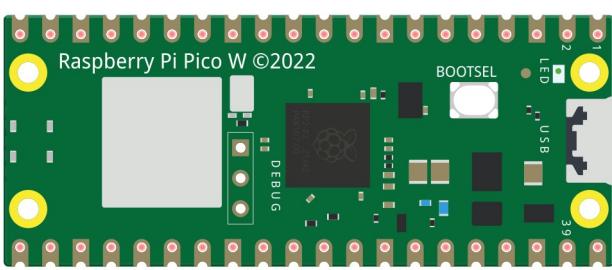
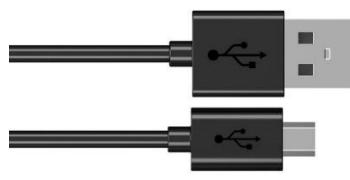
Si tiene Pico en la mano, cámbielo a Pico W antes de continuar con el aprendizaje.

En este capítulo, presentaremos cómo PICO W implementa las comunicaciones de red basadas en el protocolo TCP/IP. Hay dos roles en la comunicación TCP/IP, a saber, Servidor y Cliente, que se implementarán respectivamente con dos proyectos en este capítulo.

Proyecto 13.1 Como Cliente

En esta sección, PICO W se utiliza como Cliente para conectar el Servidor en la misma LAN y comunicarse con él.

Lista de componentes

Frambuesa Pi Pico W x1	Cable micro USB x1
	

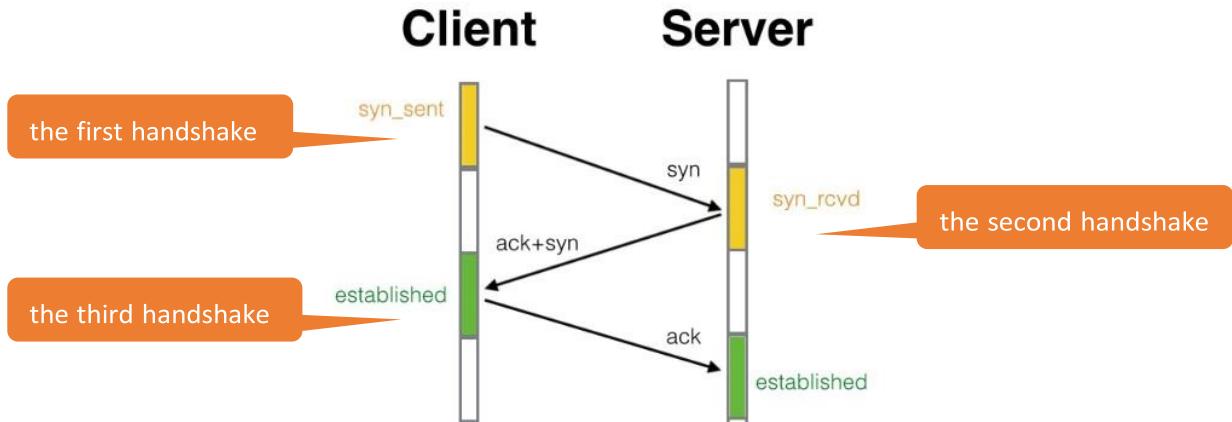
Conocimiento de componentes

conexión TCP

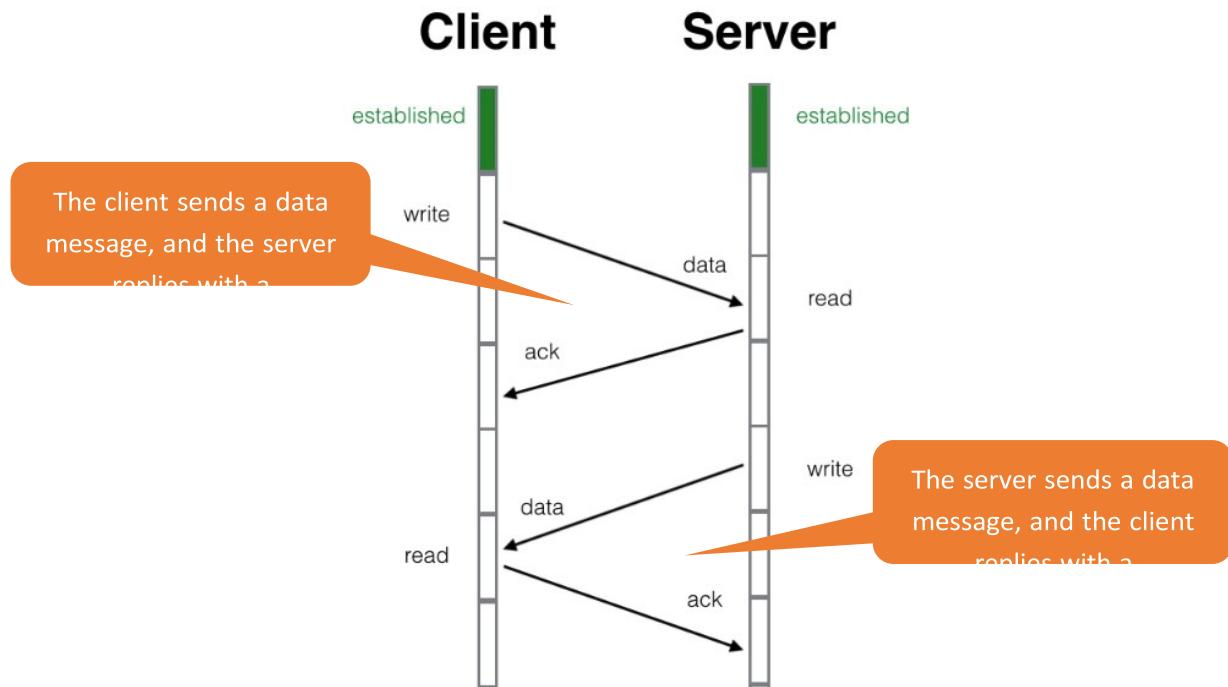
Antes de transmitir datos, TCP necesita establecer una conexión lógica entre el extremo de envío y el extremo de recepción. Proporciona una transmisión de datos confiable y sin errores entre las dos computadoras. En la conexión TCP se debe aclarar el cliente y el servidor. El cliente envía una solicitud de conexión al servidor, y cada vez que se propone dicha solicitud, se requiere un "apretón de manos de tres veces".

Apretón de manos de tres veces: en el protocolo TCP, durante la fase de preparación del envío de datos, el cliente y el servidor interactúan tres veces para garantizar la confiabilidad de la conexión, lo que se denomina "apretón de manos de tres veces". El primer apretón de manos, el cliente envía una solicitud de conexión al servidor y espera a que el servidor confirme. El segundo apretón de manos, el servidor envía una respuesta al cliente informando que ha recibido la solicitud de conexión.

El tercer apretón de manos, el cliente envía un mensaje de confirmación al servidor nuevamente para confirmar la conexión.



TCP es un protocolo de control de transmisión de bajo nivel orientado a la conexión. Después de que TCP establece una conexión, el cliente y el servidor pueden enviarse y recibir mensajes entre sí, y la conexión siempre existirá mientras el cliente o el servidor no inicien la desconexión. Cada vez que una de las partes envía un mensaje, la otra parte responderá con una señal de acuse de recibo.



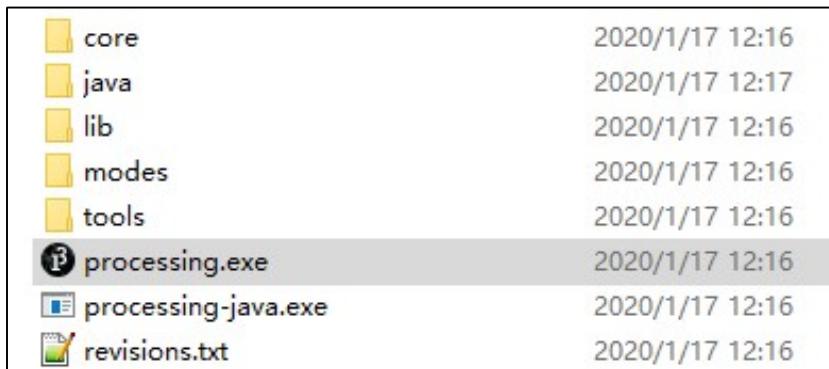
Procesamiento de instalación

En este tutorial, usamos Processing para construir una plataforma de comunicación TCP/IP simple.

Si no ha instalado Processing, puede descargarlo haciendo clic en <https://processing.org/download/>. Puede elegir una versión adecuada para descargar de acuerdo con el sistema de su PC.

The screenshot shows the official Processing website. On the left, there's a sidebar with links like Cover, Download, Donate, Exhibition, Reference, Libraries, Tools, Environment, Tutorials, Examples, Books, Overview, and People. The main content area features a large 'Processing' logo at the top. Below it, a search bar is followed by the text: "Download Processing. Processing is available for Linux, Mac OS X, and Windows. Select your choice to download the software below." To the right of this text is a circular icon with a stylized 'P'. Below the icon, the text "3.5.4 (17 January 2020)" is displayed, along with download links for Windows 64-bit, Windows 32-bit, Linux 64-bit, and Mac OS X.

Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.



Usar el modo Servidor para la comunicación

Abre el

"[Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Codes/Micropython_Codes/13.1_TCP_as_Client/sketchWiFi/sketchWiFi.pde](#)". Haga clic en "Ejecutar".

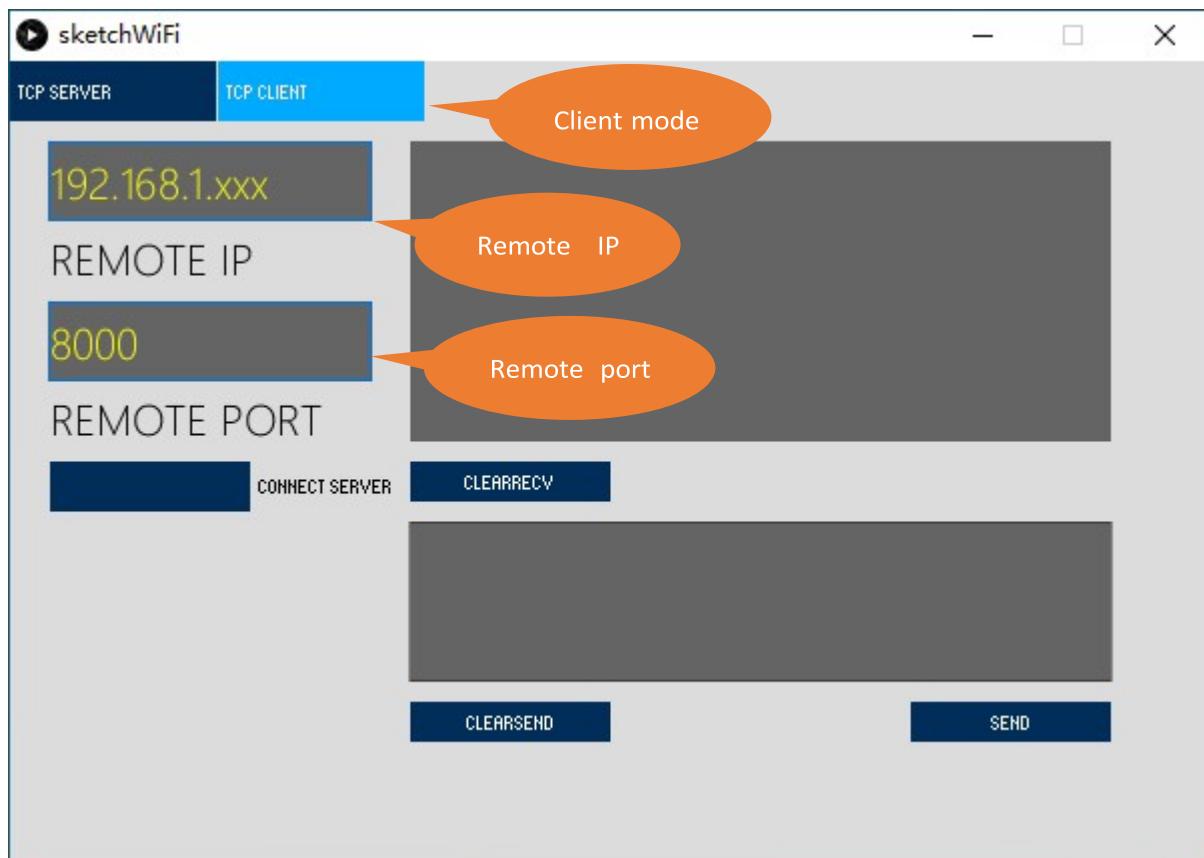


La nueva interfaz emergente es la siguiente. Si se utiliza PICO W como Cliente, seleccione el modo SERVIDOR TCP para sketchWiFi.



Cuando sketchWiFi selecciona el modo SERVIDOR TCP, el código PICO W debe cambiarse de acuerdo con la visualización de IP LOCAL o PUERTO LOCAL de sketchWiFi.

Si PICO W sirve como servidor, seleccione el modo CLIENTE TCP para sketchWiFi.



Cuando sketchWiFi selecciona el modo CLIENTE TCP, la IP LOCAL y el PUERTO LOCAL de sketchWiFi deben cambiarse de acuerdo con la dirección IP y el número de puerto impreso por el monitor serie.

modo : seleccione **Modo servidor / Modo cliente** .

Dirección IP : En el modo Servidor, no es necesario completar esta opción y la computadora obtendrá automáticamente la dirección IP.

En el modo Cliente, complete la dirección IP remota para conectarse.

Número de puerto : en el modo Servidor, complete un número de puerto para que los dispositivos cliente realicen una conexión de acceso.

En modo cliente, complete el número de puerto proporcionado por los dispositivos del servidor para realizar una conexión de acceso.

Botón de inicio : en el modo de servidor, presione el botón y luego la computadora funcionará como servidor y abrirá un número de puerto para que el cliente realice la conexión de acceso. Durante este período, la computadora seguirá monitoreando.

En modo cliente, antes de presionar el botón, asegúrese de que el servidor esté encendido, que la dirección IP remota y el número de puerto remoto sean correctos; presione el botón y la computadora realizará una conexión de acceso al número de puerto remoto de la IP remota como Cliente.

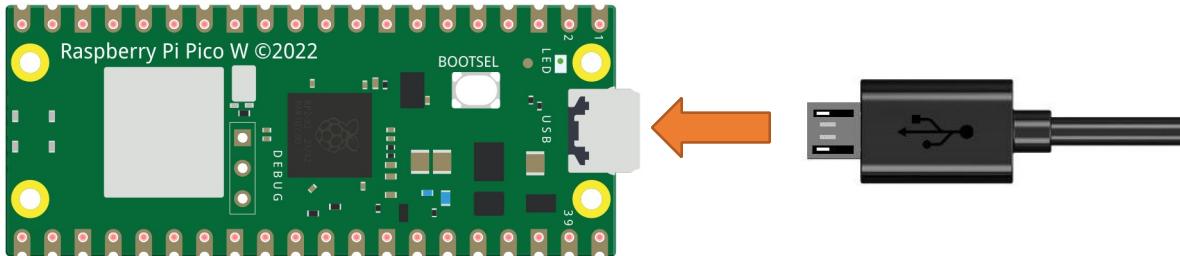
borrar recibir : borrar el contenido del cuadro de texto de recepción

borrar enviar : borrar el contenido del cuadro de texto de envío

Botón de envío : presione el botón de envío, la computadora enviará el contenido en el cuadro de texto a otros.

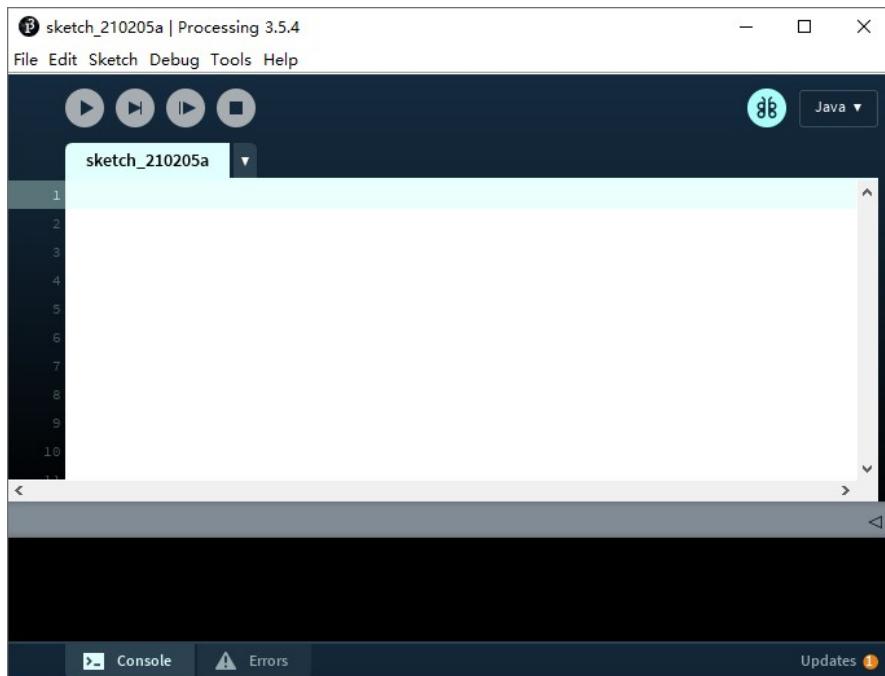
Circuito

Connect Pico W to the computer using the USB cable.

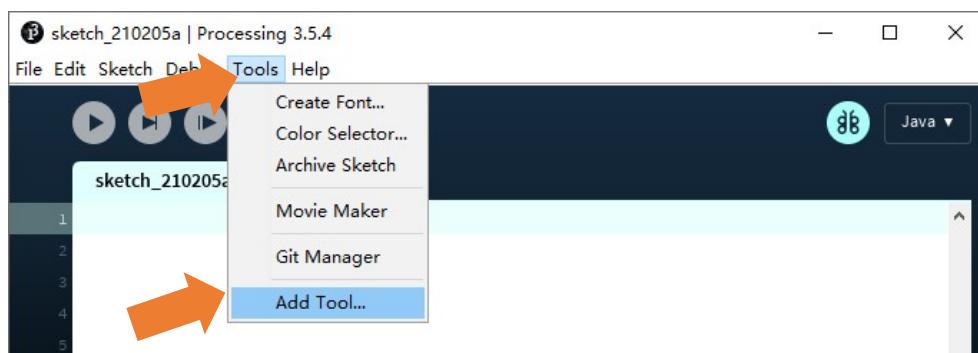


Código

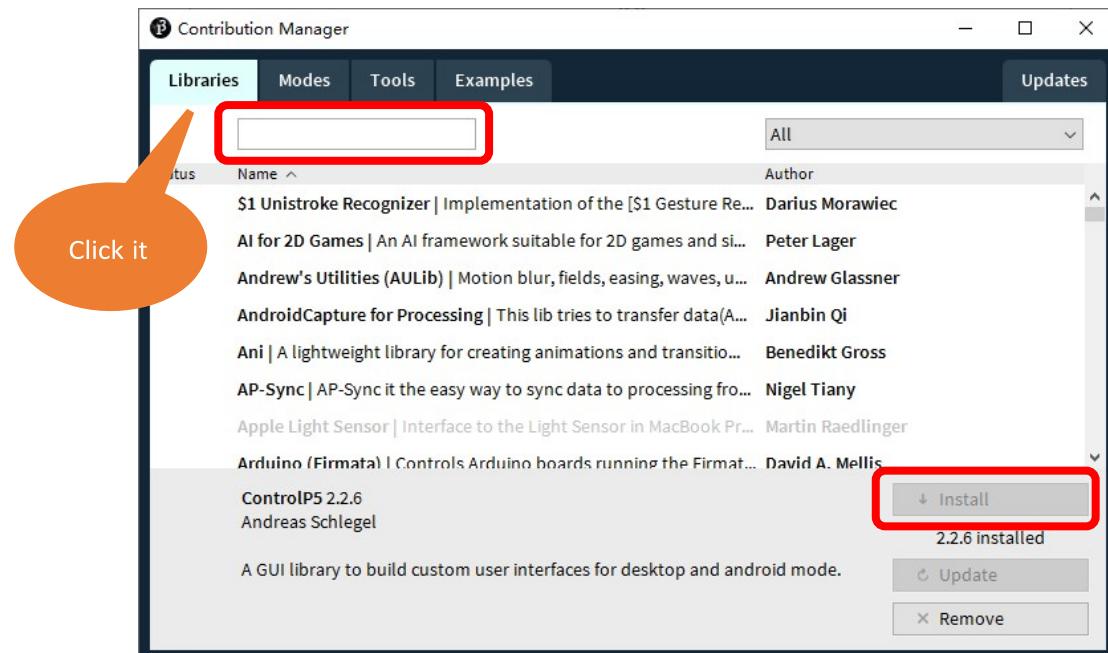
If you have not installed "ControlP5", follow the following steps to continue with the installation; if you have installed it, skip this section. Open the sketch.



Click Add Tool under Tools.

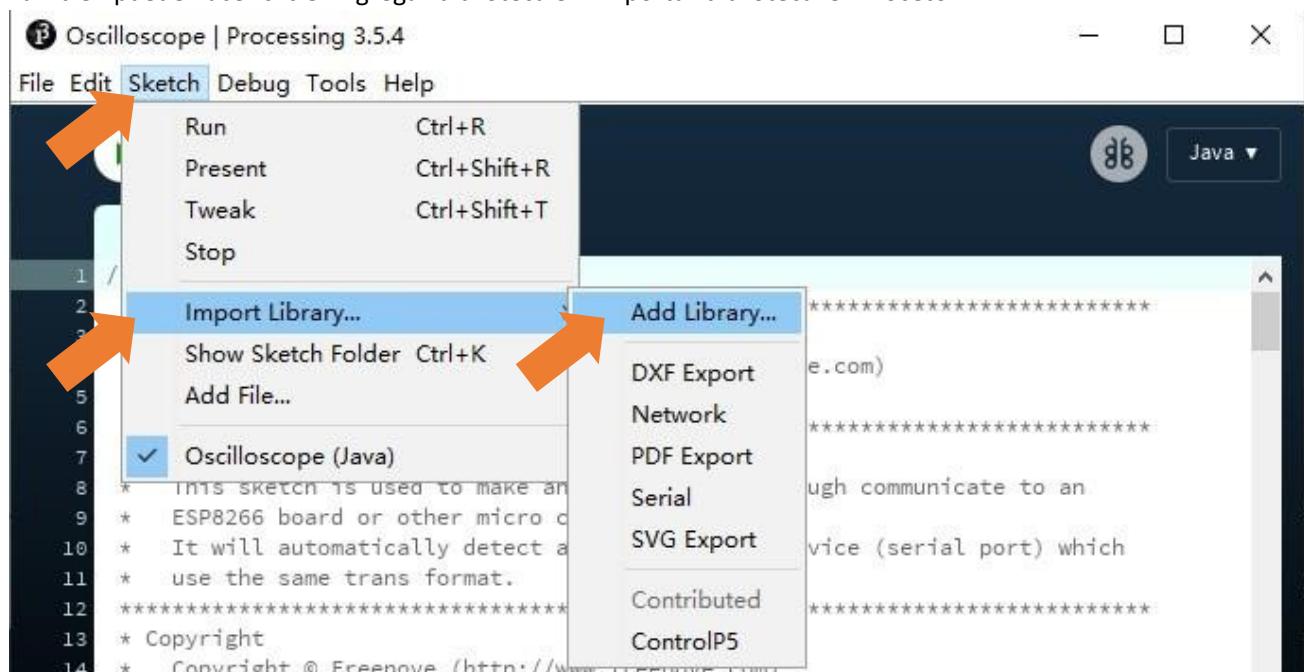


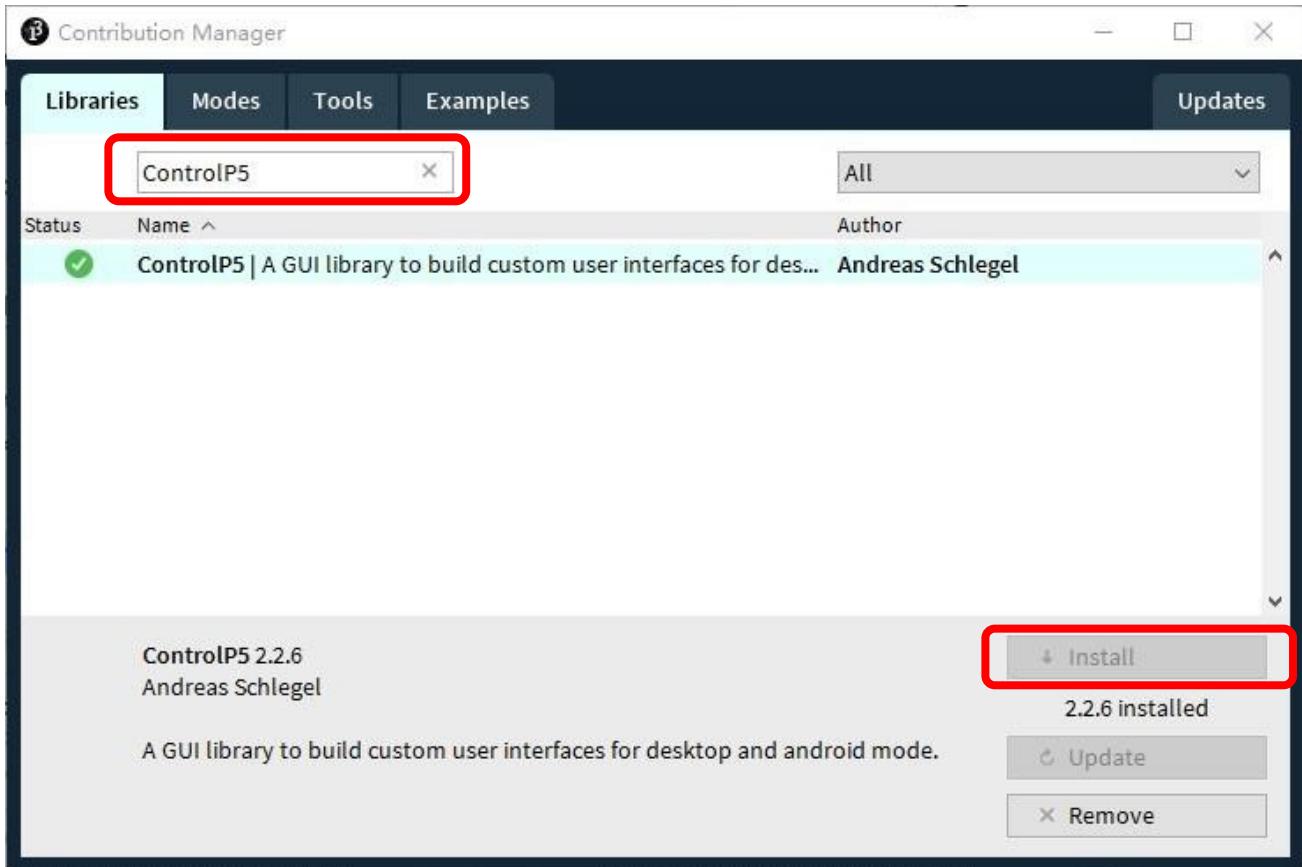
Select Libraries in the pop-up window.



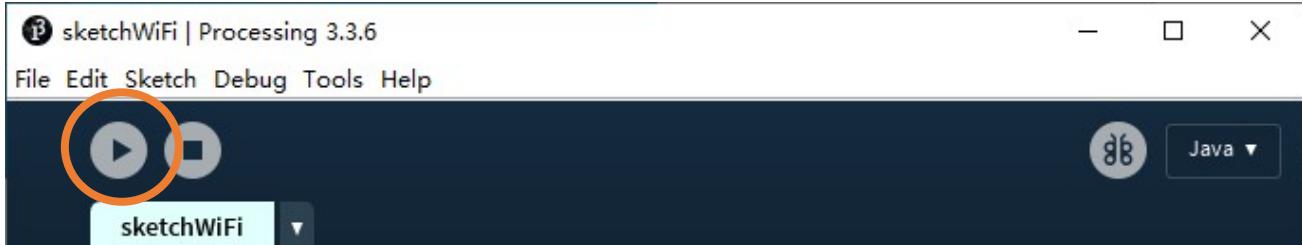
Ingrese "ControlP5" en el cuadro de búsqueda y luego seleccione la opción que se muestra a continuación. Haga clic en "Instalar" y espere a que finalice la instalación.

También puede hacer clic en Agregar biblioteca en 'Importar biblioteca' en 'Boceto'.





Antes de ejecutar el Código, abra "sketchWiFi.pde". primero, y haga clic en "Ejecutar".



La nueva ventana emergente utilizará la dirección IP de la computadora de forma predeterminada y abrirá un puerto de monitor de datos. Haga clic en "Escuchar".



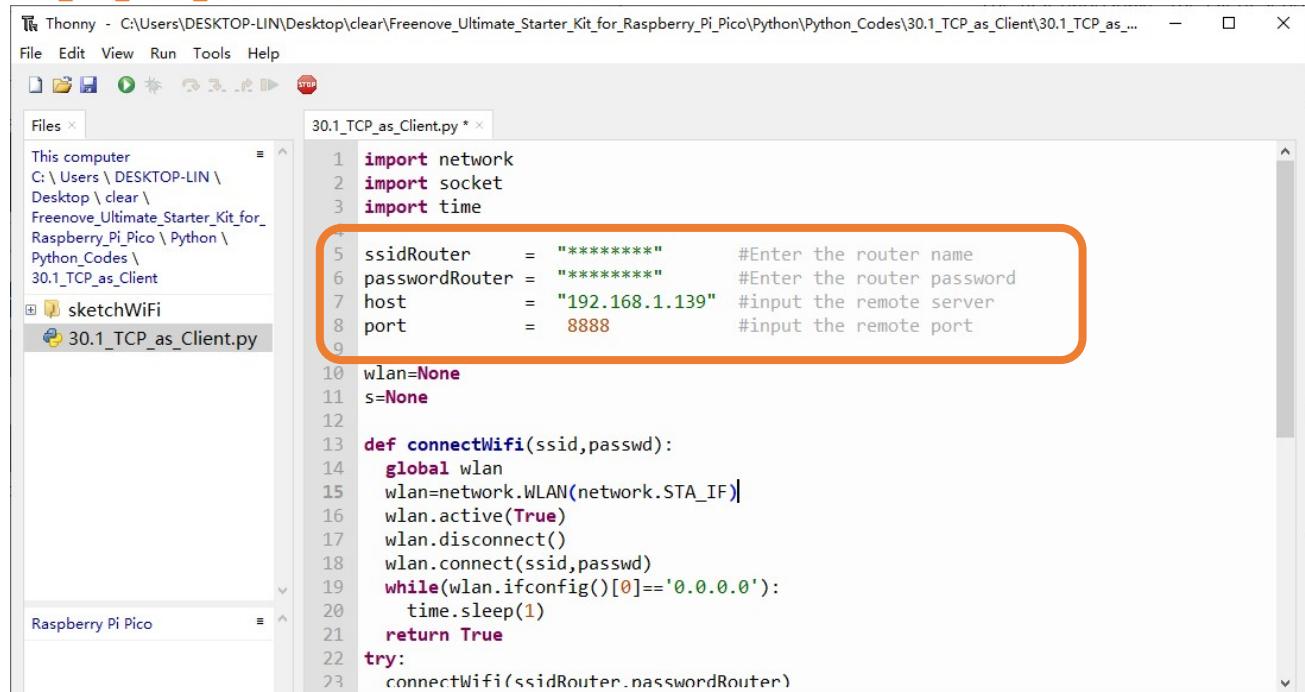
Capítulo 13 TCP/IP (solo para Pico W)

Mueva la carpeta del programa " Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes " al disco (D) de antemano con la ruta de " D:/Micropython_Codes ".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "13.1_TCP_as_Client" y haga doble clic en "13.1_TCP_as_Client.py".

Antes de hacer clic en "Ejecutar secuencia de comandos actual", modifique el nombre y la contraseña de su enrutador y complete el "host" y el "puerto" de acuerdo con la **información de IP en la aplicación de procesamiento que** se muestra en el cuadro a continuación:

13.1_TCP_como_cliente

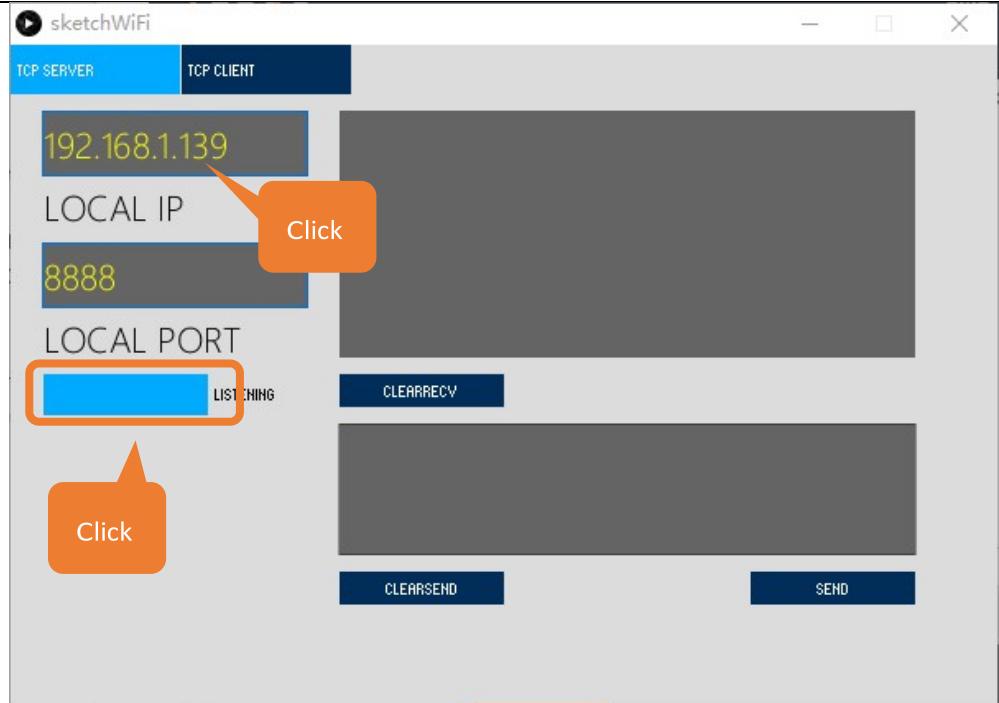


```

1 import network
2 import socket
3 import time
4
5 ssidRouter      = "*****"           #Enter the router name
6 passwordRouter = "*****"           #Enter the router password
7 host            = "192.168.1.139"    #input the remote server
8 port             = 8888              #input the remote port
9
10 wlan=None
11 s=None
12
13 def connectWifi(ssid,passwd):
14     global wlan
15     wlan=network.WLAN(network.STA_IF)
16     wlan.active(True)
17     wlan.disconnect()
18     wlan.connect(ssid,passwd)
19     while(wlan.ifconfig()[0]=='0.0.0.0'):
20         time.sleep(1)
21     return True
22 try:
23     connectWifi(ssidRouter,passwordRouter)

```

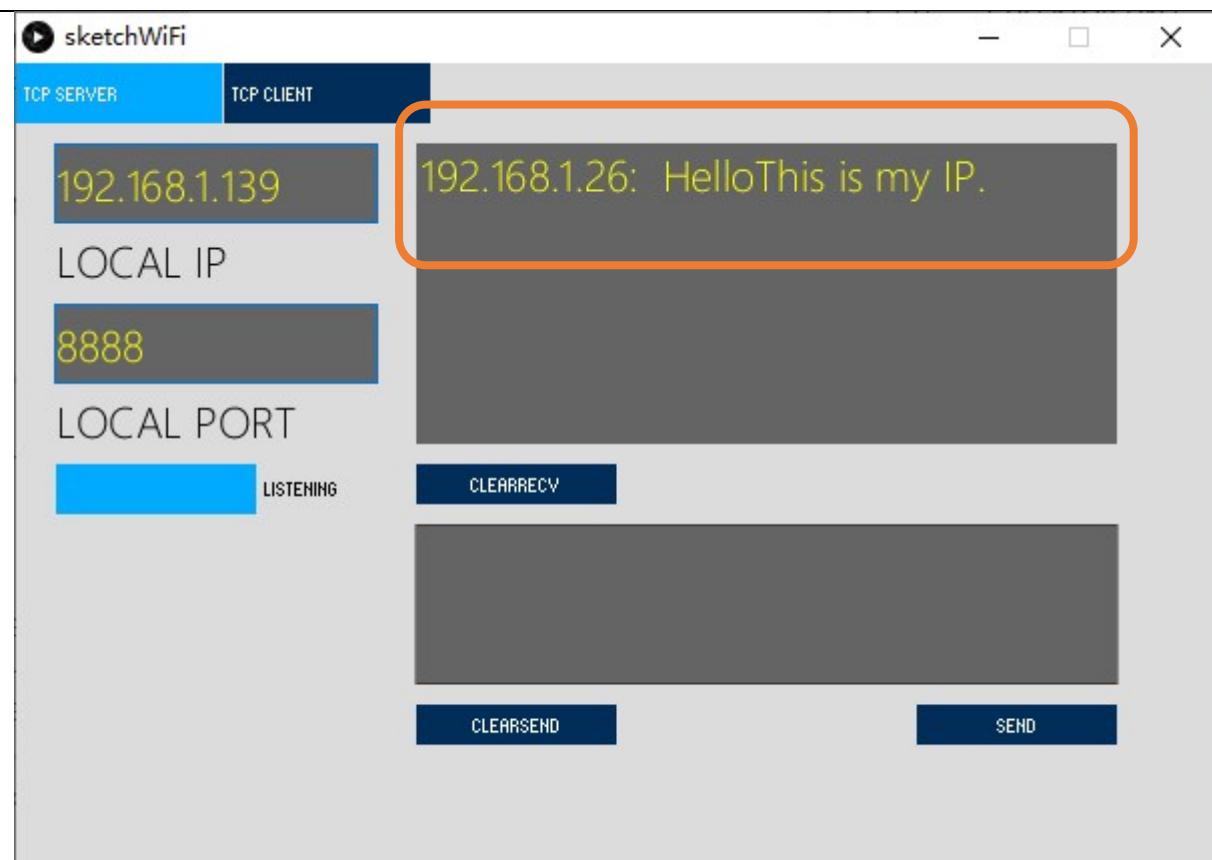
Haga clic en "Ejecutar script actual" y en "Shell", puede ver que PICO W se conecta automáticamente a sketchWiFi.



Si no hace clic en "Escuchar" para sketchWiFi, PICO W no podrá conectarse e imprimirá la información de la siguiente manera:

```
Shell >>>
>>>
MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
    TCP close, please reset!
>>>
```

PICO W se conecta con TCP SERVER, y TCP SERVER recibe mensajes de PICO W, como se muestra en la figura a continuación.



En este punto, puede enviar datos a Pico W a través de sketchWiFi. Pico W enviará los datos recibidos a sketchWiFi después de recibirlos.

El siguiente es el código del programa:

```
1 importar red
2 importar socket
3 tiempo de
4 importación
5
6 ssidRouter = "*****" #Ingrese el nombre del
7 enrutador passwordRouter = "*****" #Ingrese la
8 contraseña del enrutador host = "*****" #ingrese el
9 control remoto puerto del servidor = 8888 #ingrese el
puerto remoto
```

```

10 wlan= Ninguno s=
11 Ninguno
12 definitivamente
13 conectarWifi ( ssid ,
14 contraseña):
15     wlan global wlan=
diecisí  network.WLAN(network.STA_IF)
s         wlan.active( True )
17     wlan.disconnect()
18     wlan.connect(ssid,passwd) while
19     (wlan.ifconfig()[0]== '0.0.0.0'
20 ) :
21     tiempo.dormi
22     r(1) volver
23 Verdadero
24 prueba :
25 connectWifi(ssidRouter,passwordRoute
26 r) s = socket.socket()
27 s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
28 s.connect((host,puerto)) print ( "TCP
29 conectado a:" , host, ":" , puerto)
30 s.send( 'Hola' )
31 s.send( 'Esta es mi IP.' )
32 mientras cierto :
33 datos = s.recv(1024) if ( len
34 (datos) == 0):
35     imprimir ( "Cerrar socket" )
36 s.close()
37 descanso
38 imprimir
39 (datos)
40 ret=s.send(dato
41 s) excepto :
42     print ( "Cierre TCP,
43 reinicie!" ) if (s):
        s.close()
        wlan.disconnect()
        wlan.active( Falso )

```

Importar módulos de red, socket y tiempo.

1	importar red
2	importar socket
3	tiempo de
	importación

Ingrese el nombre real del enrutador, la contraseña, la dirección IP del servidor remoto y el número de puerto.

```

5   ssidRouter = "*****" #Ingrese el nombre del enrutador
6   passwordRouter = "*****" #Ingrese la contraseña del
7   enrutador
8   host = "*****" #ingrese el control remoto
puerto del servidor = 8888 #ingrese el puerto remoto

```

Conecte el enrutador especificado hasta que tenga éxito.

```

13  definitivamente conectarWifi ( ssid , contraseña ) :
14      wlan global wlan =
15      network.WLAN ( network.STA_IF )
dieciséis wlan.active ( True )
17      wlan.disconnect ()
18      wlan.connect ( ssid , passwd ) while
19          ( wlan.ifconfig () [ 0 ] == ' 0.0.0.0 ' )
20      ) :
21      tiempo.dormir ( 1 )
volver Verdadero

```

Conecte el enrutador y luego conéctelo al servidor remoto.

```

23  connectWifi ( ssidRouter , passwordRouter ) s =
24  socket.socket ()
25  s.setsockopt ( socket.SOL_SOCKET , socket.SO_REUSEADDR , 1 )
26  s.connect (( host , puerto )) print ( "TCP
27  conectado a:" , host , ":" , puerto )

```

Envíe mensajes al servidor remoto, reciba los mensajes de él e imprimalos, y luego envíe los mensajes de vuelta al servidor.

```

28  s.send ( 'Hola' )
29  s.send ( 'Esta es mi IP.' )
30  mientras cierto :
31  datos = s.recv ( 1024 ) if ( len ( datos )
32  == 0 ) :
33      imprimir ( "Cerrar socket" )
34  s.close () romper
35  imprimir ( datos )
36  ret = s.send ( datos )
37

```

Si ocurre una excepción en el programa, por ejemplo, el servidor remoto se apaga, ejecute el siguiente programa, apague la función de enchufe y desconecte el WiFi.

```

39  print ( "Cierre TCP, reinicie!" ) if
40  ( s ) :
41  s.close () wlan.disconnect ()
42  wlan.active ( Falso )
43

```

Referencia

Zócalo de clase

Antes de cada uso de **socket**

socket.socket(**tipos**, **af**: d, **socket**, agregue la instrucción " **import socket**" en la parte superior del archivo **socket.AF_INET** **socket.socket**.**AI** **python. proto]) : Crea un socket.**

tipos : tipos

: IPv4

: IPv6

zócalo.SOCK_RAW

CK_STREAM : flujo TCP

proto : protocolo **CK_DGRAM** : datagrama UDP

: Enchufe original

_REUSEADDR : número de

socket.setsockopt(nivel) socket reutilizable

'**SO_RCVTIMEO**' : TCPmode **socket.IPPROTO_UDP** : UDPmode

nombreopción : O **value**) : Configure el socket según las opciones.

socket. de nivel de socket

val_SOCKET : Opción de nivel de socket. Por defecto, es

socket.connect(dirección) : 95. opciones de enchufe

enviar(bytes) : **SO_REUSEADDR** : Permitir que una interfaz de socket se vincule a una dirección que **recv(bufsize)** : **close()** : ya está en uso.

puede ser un número entero o un objeto similar a bytes

que representa un búfer. : Para conectarse al servidor.

o lista de la dirección del servidor y el número

de puerto finaliza los datos y devuelve los

bytes enviados.

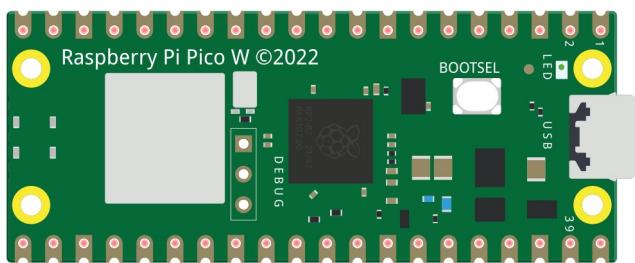
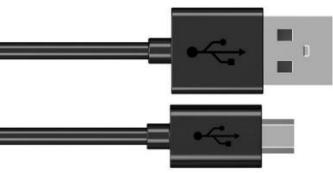
Recibe datos y devuelve un objeto de bytes que representa los datos os. Cerrar enchufe.

Para obtener más información, visite: <http://docs.micropython.org/en/latest/>

Proyecto 13.2 como servidor

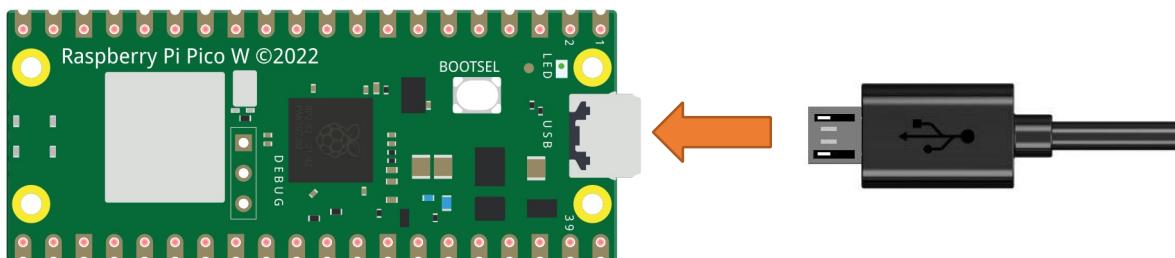
En esta sección, PICO W se utiliza como Servidor para esperar la conexión y comunicación con el Cliente en la misma LAN.

Lista de componentes

Frambuesa Pi Pico W x1	Cable micro USB x1
	

Círculo

Connect Pico W to the computer using the USB cable.



Código

Mueva la carpeta del programa " **Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes** " al disco (D) de antemano con la ruta de " **D:/Micropython_Codes** ".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "13.2_TCP_as_Server" y haga doble clic en "13.2_TCP_as_Server.py".

Antes de hacer clic en "Ejecutar secuencia de comandos actual", modifique el nombre y la contraseña de su enrutador que se muestran en el cuadro a continuación.

13.2_TCP_as_Server

The screenshot shows the Thonny IDE interface. In the center, the code editor displays the file `30.2_TCP_as_Server.py`. A red box highlights the configuration section:

```

1 import network
2 import socket
3 import time
4
5 ssidRouter      = "*****"           #Enter the router name
6 passwordRouter = "*****"           #Enter the router password
7 port            = 8000               #input the remote port
8 wlan=None
9 listenSocket=None
10
11 def connectWifi(ssid,passwd):
12     global wlan
13     wlan=network.WLAN(network.STA_IF)
14     wlan.active(True)
15     wlan.disconnect()
16     wlan.connect(ssid,passwd)
17     while(wlan.ifconfig()[0]!='0.0.0.0'):
18         print ('Connect to a wireless network...')
19         time.sleep(1)

```

In the bottom right corner of the code editor, it says "MicroPython (Raspberry Pi Pico)".

Después de asegurarse de que el nombre y la contraseña del enrutador son correctos, haga clic en "Ejecutar script actual" y en "Shell", puede ver un servidor abierto por el PICO W esperando conectarse a otros dispositivos de red.

```
Shell x
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
tcp: waiting...
Server IP: 192.168.1.26      Port: 8000
tcp: accepting.....
Close TCP-Server, please reset.

>>>
```

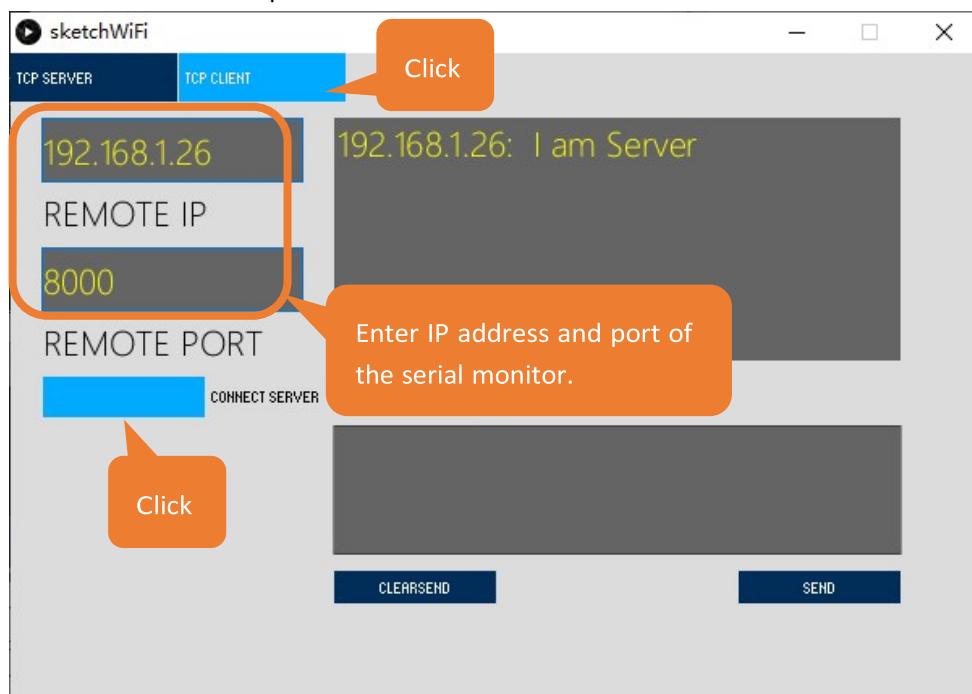
IP address and port

Procesando:

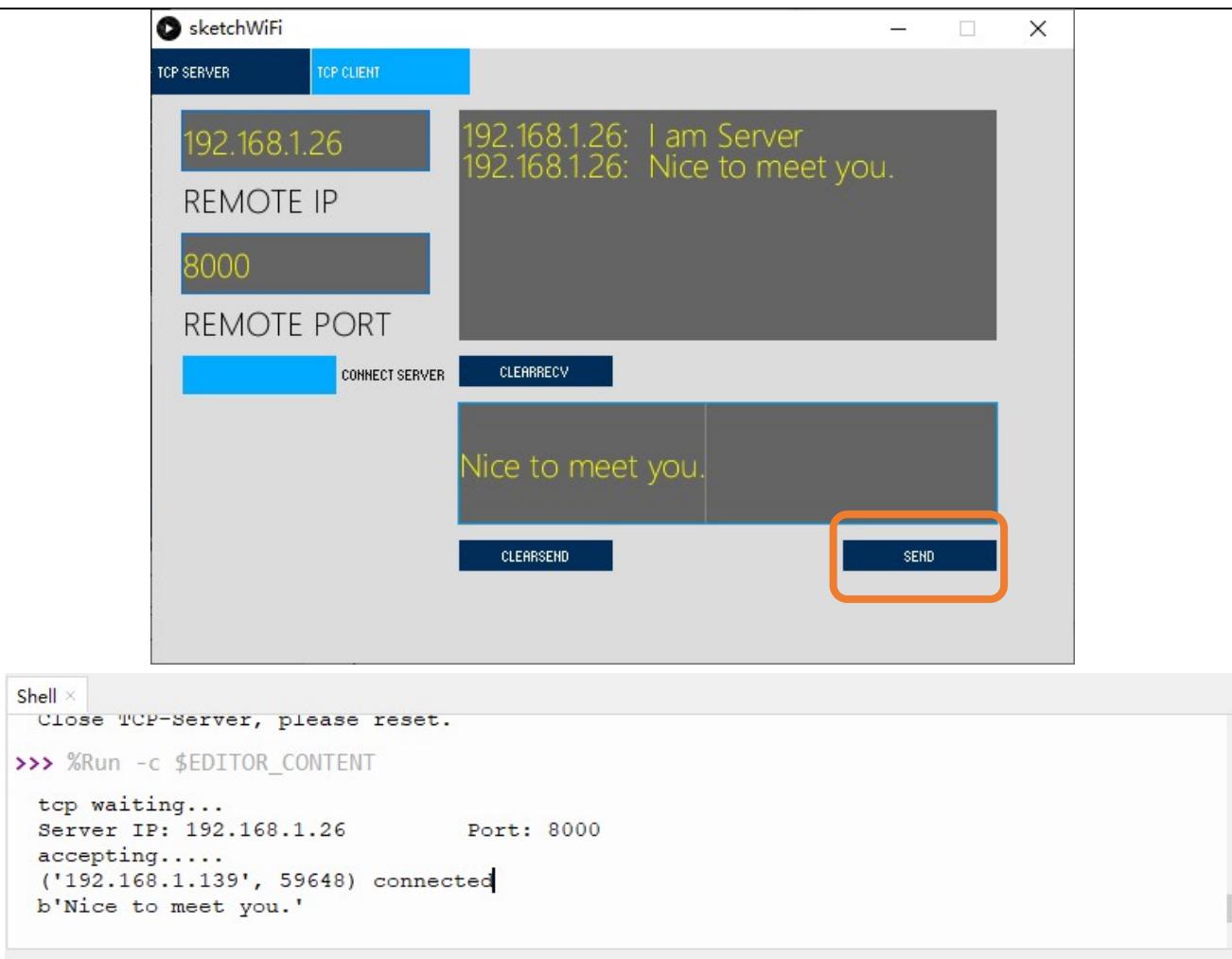
Abre el

"**Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Codes/MicroPython_Codes/13.2_TCP_as_Server/sketchWiFi/sketchWiFi.pde**".

Según el mensaje impreso en "Shell", ingrese la dirección IP y el puerto correctos al procesar, y haga clic para establecer una conexión con PICO W para comunicarse.



Puede ingresar cualquier información en el "Cuadro de envío" de sketchWiFi. Haga clic en "Enviar" y PICO W imprimirá los mensajes recibidos en "Shell" y los enviará de vuelta a sketchWiFi.



El siguiente es el código del programa:

```
1 importar red
2 importar
3 socket tiempo
4 de importación
5 ssidRouter = "*****" #Ingrese el nombre del
6 enrutador passwordRouter = "*****" #Ingrese la
7 contraseña del enrutador port = 8000 #ingrese el puerto
8 remoto wlan = Ninguno listenSocket = Ninguno
9 definitivamente
10 conectarWifi ( ssid ,
11 contraseña):
12     wlan global
13     wlan=network.WLAN(network.STA_IF)
14     wlan.active( True )
15     wlan.disconnect()
16 dieciséi wlan.connect(ssid,passwd) while
17 s (wlan.ifconfig()[0]== '0.0.0.0'
18 ):
19     tiempo.dormir(1)
20     volver Verdadero
21
22 pru
23 eba
24 :
25 connectWifi(ssidRouter,passwordRouter) ip=wlan.ifconfig()[0]
26 listenSocket = socket.socket() listenSocket.bind((ip,puerto))
27 listenSocket.listen(1)
28 listenSocket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR
29 , 1) imprimir ( 'tcp esperando...' ) mientras cierto :
30     print ( "IP del servidor:" ,ip,
31 "\tPort:" ,port) print (
32 "aceptando..." ) conn,addr =
33 listenSocket.accept() print (addr,
34 "conectado" ) break conn.send (
35 'Soy servidor' )
```

```
36     tiempo cierto :
37     datos = conn.recv(1024) if ( len (datos)
38     == 0) :
39         print ( "cerrar
40         socket" )
41     listenSocket.close()
42     wlan.disconnect()
43     wlan.active( False ) romper
44     más :
45         imprimir (datos) ret
= conn.send (datos)
```

46

Capítulo 13 TCP/IP (solo para Pico W)

```

36     tiempo cierto :
37     datos = conn.recv(1024) if ( len (datos)
38     == 0):
39         print ( "cerrar
40 socket" )
41     listenSocket.close()
42     wlan.disconnect()
43     wlan.active( False ) romper
44 más :
45     imprimir (datos) ret
46     = conn.send (datos) excepto
47     :
48         print ( "Cierre el servidor TCP,
49 reinicie". ) if (listenSocket):
50     listenSocket.close() wlan.disconnect()
51     wlan.active( False )
52

```

Llame a la función connectWifi() para conectarse al enrutador y obtener la IP dinámica que asigna a PICO W.

```

22     connectWifi(ssidRouter, passwordRouter) ip=wlan.ifconfig()[0]
23

```

Abra el servidor de socket, vincule el servidor a la IP dinámica y abra un puerto de monitoreo de datos.

```

24     listenSocket = socket.socket() listenSocket.bind((ip,puerto))
25     listenSocket.listen(1) listenSocket.setsockopt(socket.SOL_SOCKET,
26     socket.SO_REUSEADDR, 1)
27

```

Imprima la dirección IP y el puerto del servidor, controle el puerto y espere la conexión de otros dispositivos de red.

```
29     tiempo cierto :
30         print ( "IP del servidor:" , ip,
31             "\tPort:" , port) print ( "aceptando..." )
32         conn, addr = listenSocket.accept() print
33             (addr, "conectado" ) break
34
```

Cada vez que reciba datos, imprímalos en "Shell" y envíelos de vuelta al cliente.

Si el cliente está desconectado, cierre el servidor y desconecte WiFi.

```
47     excepto :
48         print ( "Cierre el servidor TCP,
49             reinicie." ) if (listenSocket):
50             listenSocket.close() wlan.disconnect()
51             wlan.active( False )
52
```

Capítulo 14 LED de control con Web (solo para Pico W)

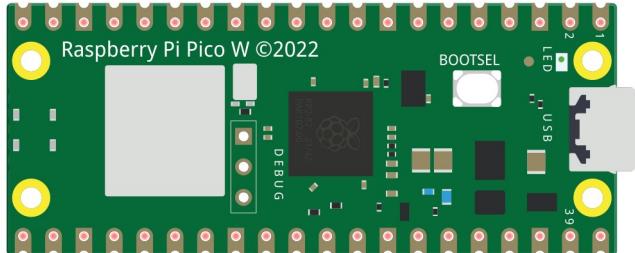
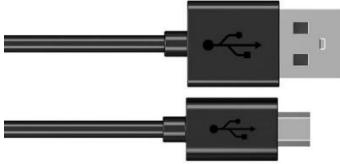
Si tiene Pico en la mano, cámbielo a Pico W antes de continuar con el aprendizaje.

En este capítulo, usaremos PICO W para hacer una casa inteligente simple. Aprenderemos a controlar luces LED a través de páginas web.

Proyecto 14.1 Control_LED_a_través_de_Web

En este proyecto, necesitamos construir un servicio web y luego usar PICO W para controlar el LED a través del navegador web del teléfono o PC. A través de este ejemplo, puede controlar de forma remota los electrodomésticos de su hogar para lograr un hogar inteligente.

Lista de componentes

Frambuesa Pi Pico W x1	Cable micro USB x1
	

Conocimiento de componentes

HTML

El lenguaje de marcado de hipertexto (HTML) es un lenguaje de marcado estándar para crear páginas web. Incluye un conjunto de etiquetas que unifican documentos en la red y conectan recursos dispares de Internet en un todo lógico. El texto HTML es un texto descriptivo compuesto por comandos HTML que describen el texto, , gráficos, animaciones, sonidos, tablas, enlaces, etc. La extensión del archivo HTML es HTM o HTML. El hipertexto es una forma de organizar la información. Utiliza hipervínculos para asociar palabras y gráficos en texto con otros medios de información. Los medios de información pueden estar en el mismo Texto, otros archivos o archivos ubicados en una computadora remota. Esta forma de organizar la información conecta los recursos de información distribuidos en diferentes lugares, lo cual es conveniente para que las personas busquen y recuperen información.

La naturaleza de la Web es el Lenguaje de marcado de hipertexto (HTML), que puede combinarse con otras tecnologías Web (por ejemplo, lenguajes de secuencias de comandos, interfaces de puerta de enlace comunes, componentes, etc.) para crear potentes páginas Web. Así, el Lenguaje de Marcado de HIPERtexto (HTML) es la base de la programación de la World Wide Web (Web), es decir, la World Wide Web se basa en el hipertexto. El lenguaje de marcado de hipertexto se denomina lenguaje de marcado de hipertexto porque el texto contiene los llamados puntos de "hipervínculo".

Puede crear su propio sitio WEB utilizando HTML, que se ejecuta en el navegador y es analizado por el navegador. El análisis de ejemplo se muestra en la siguiente figura:



<!DOCTYPE html>: Declararlo como un documento HTML5

<html>: es el elemento raíz de una página HTML

<head>: contiene metadatos del documento, como < juego de caracteres meta="utf-8"> Defina el formato de codificación de la página web en UTF-8. **<título>**: Notasel título del documento

<cuerpo>: contiene contenido de página visible

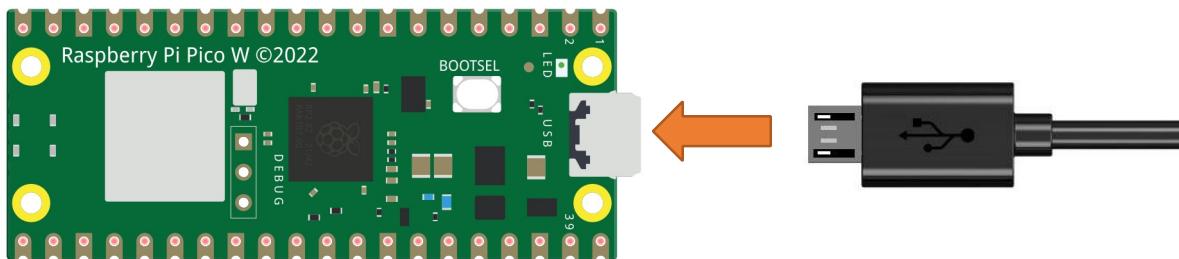
<h1>: Definir un encabezado grande

<p>: Definir un párrafo

Para obtener más información, visite: <https://developer.mozilla.org/en-US/docs/Web/HTML>

Circuito de control LED con web (solo para Pico W)

Connect Pico W to the computer using the USB cable.

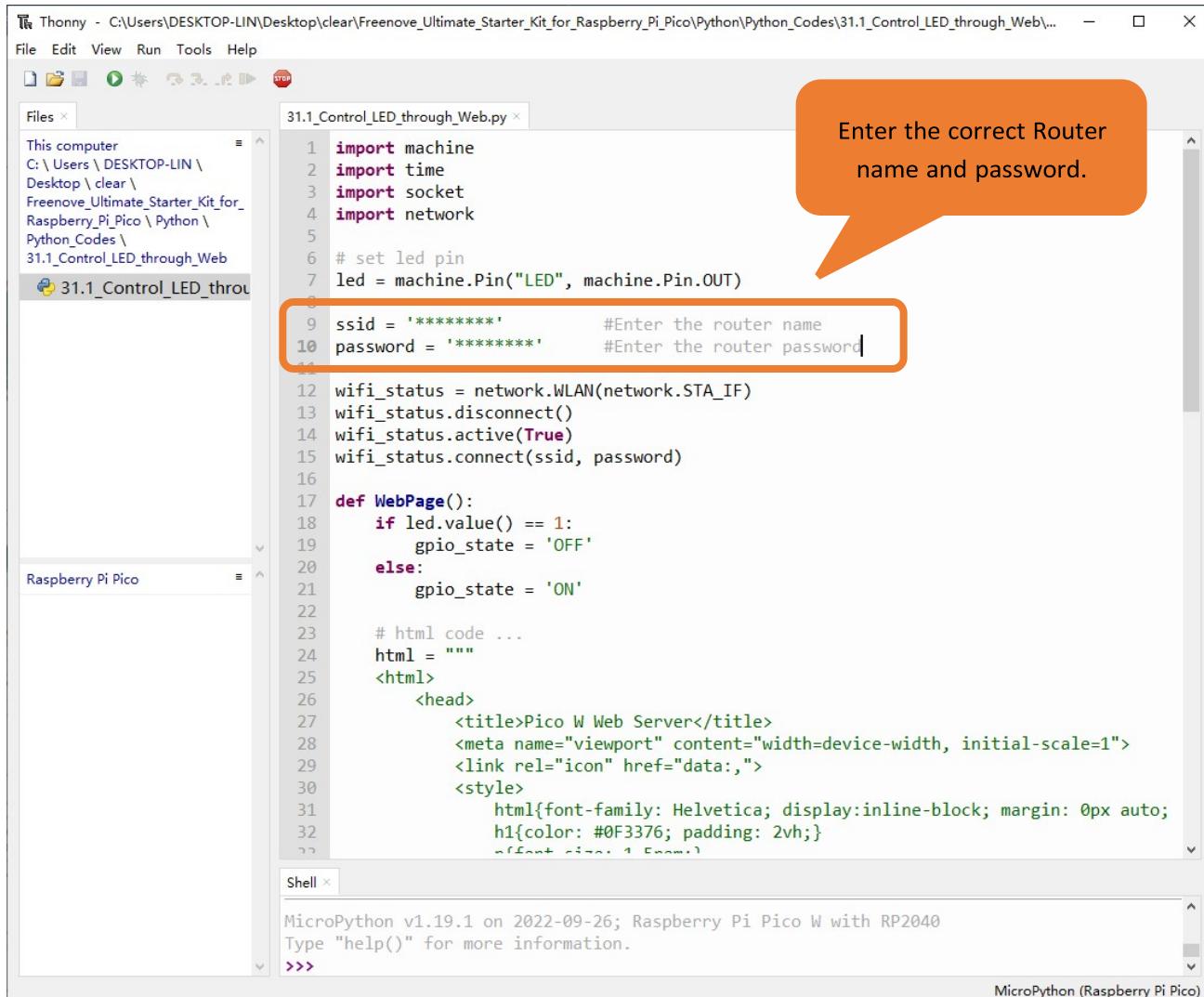


Código

Mueva la carpeta del programa "**Freenove_Basic_Starter_Kit_for_Raspberry_Pi_Pico/Python/Python_Codes**" al disco (D) de antemano con la ruta de "**D:/Micropython_Codes**".

Abra "Thonny", haga clic en "Esta computadora" → "D:" → "Micropython_Codes" → "14.1_Control_LED_mediante_Web". y haga doble clic en "14.1_Control_LED_through_Web.py".

14.1_Control_LED_through_Web



The screenshot shows the Thonny IDE interface with the file `31.1_Control_LED_through_Web.py` open. The code imports machine, time, socket, and network modules. It sets up a LED pin and defines a function `WebPage()` that checks the state of the LED and generates an HTML response. Two lines of code, `ssid = '*****'` and `password = '*****'`, are highlighted with a red box and a callout bubble containing the text: "Enter the correct Router name and password."

```

import machine
import time
import socket
import network

# set led pin
led = machine.Pin("LED", machine.Pin.OUT)

ssid = '*****'          #Enter the router name
password = '*****'      #Enter the router password

wifi_status = network.WLAN(network.STA_IF)
wifi_status.disconnect()
wifi_status.active(True)
wifi_status.connect(ssid, password)

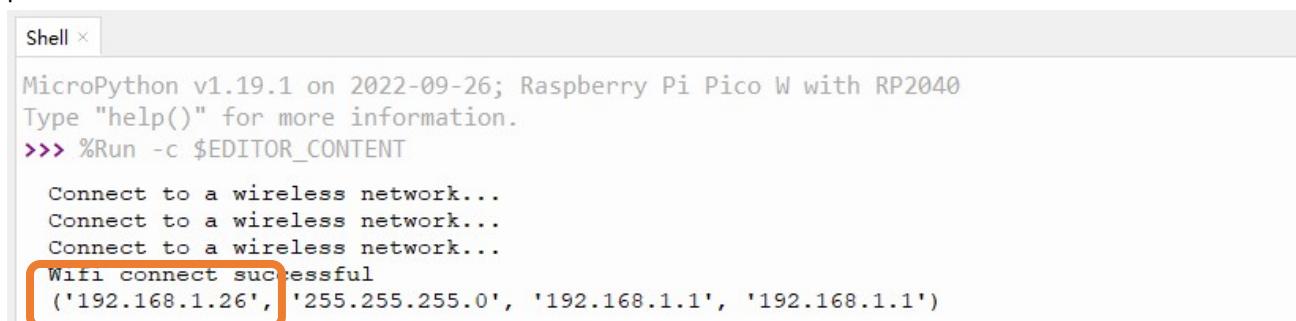
def WebPage():
    if led.value() == 1:
        gpio_state = 'OFF'
    else:
        gpio_state = 'ON'

    # html code ...
    html = """
<html>
<head>
    <title>Pico W Web Server</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="icon" href="data:,>
    <style>
        html{font-family: Helvetica; display:inline-block; margin: 0px auto;
        h1{color: #0F3376; padding: 2vh;}
</head>
<body>
    <h1>Pico W Web Server</h1>
    <p>GPIO State: <strong>{gpio_state}</strong></p>
</body>
</html>
    """.format(gpio_state=gpio_state)
    return html

```

Debido a que los nombres y contraseñas de los enrutadores en varios lugares son diferentes, antes de que se ejecute el Código, los usuarios deben ingresar el nombre y la contraseña correctos del enrutador en el cuadro como se muestra en la ilustración anterior.

Después de asegurarse de que el nombre y la contraseña del enrutador se ingresaron correctamente, compile y cargue los códigos en PICO W, espere a que PICO W se conecte a su enrutador e imprima la dirección IP asignada por el enrutador a PICO W en "Shell".



The screenshot shows the MicroPython shell output. It starts with the version information: "MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040". It then shows the command `>>> %Run -c $EDITOR_CONTENT`. The output indicates the device is connecting to a wireless network, and finally, it shows a successful connection message: "Wifi connect successful ('192.168.1.26', '255.255.255.0', '192.168.1.1', '192.168.1.1')".

```

MicroPython v1.19.1 on 2022-09-26; Raspberry Pi Pico W with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

Connect to a wireless network...
Connect to a wireless network...
Connect to a wireless network...
Wifi connect successful
('192.168.1.26', '255.255.255.0', '192.168.1.1', '192.168.1.1')

```

Cuando PICO W se conecta con éxito a "ssid", "Shell" muestra la dirección IP asignada a PICO W por el enrutador. Acceda a <http://192.168.1.26> en un navegador de computadora en la LAN. Como se muestra en la siguiente figura:

Capítulo 14 LED de control con Web (solo para Pico W)



Puede hacer clic en el botón correspondiente para controlar el encendido y apagado del LED. El siguiente es el código del programa:

```

1 Importar
2 máquina
3 Importar
4 tiempo
5 Importar
6 enchufe
7 Importar red
8
9 # establecer led pin led =
10 máquina.Pin( "LED",
11 máquina.Pin.OUT)
12 ssid = '*****'           #Ingrese la
13 contraseña del nombre del enrutador =
14 '*****'                 #Ingrese la contraseña del
15 enrutador
16
17 diecisí
18 s
19 wifi_status = network.WLAN(network.STA_IF)
20 wifi_status.disconnect()
21 wifi_status.active( True )
22 wifi_status.connect(ssid, contraseña)
23
24
25 defini
26 vamente
27 Página
28 web () :
29     si led.value() == 1 :
30         gpio_state = 'Activado'
31     más :

```

```
gpio_state = 'DESACTIVADO'

# código HTML ...
html = """
<html>
<cabeza>
<title>Servidor web Pico W</title>
```


Capítulo 14 LED de control con Web (solo para Pico W)

```

69         imprimir ( 'LED
70 ENCENDIDO' ) led.valor( 1 )
71 más :
72         imprimir ( 'LED
73 APAGADO' ) led.valor( 0 ) si
74 led.valor() == 1 : gpio_state
75 = 'Encendido'     más :
76 gpio_state = 'OFF' respuesta = WebPage()
77 conn.send( 'HTTP/1.1 200 OK \n' ) conn.send(
78 'Content-Type: text/html \n' ) conn.send(
79 'Conexión: cerrar \n \n' )
80 conn.sendall(respuesta) conn.close() excepto :
81 conn.close()
82 s.close() pasar
83
84
85
86
87

```

Importar módulo de socket e Importar módulo de red.

```

3 importar socket
4 importar red

```

Ingresel nombre de AP y la contraseña correctos.

```

3 ssid = '*****'          #Ingrese la contraseña del
4 nombre del enrutador = '*****'      #Ingrese la
contraseña del enrutador

```

Configure PICO W en modo Estación y conéctelo a su enrutador.

```

12 wifi_status = network.WLAN(network.STA_IF)
13 wifi_status.disconnect() wifi_status.active( True
14 ) wifi_status.connect(ssid, contraseña)
15

```

"Shell" muestra la dirección IP asignada a PICO W.

```

53 imprimir (wifi_status.ifconfig())

```

Haga clic en el botón de la página web para controlar el encendido y apagado de la luz LED.

```
68         if led_on == 6: print
69             ( 'LED ON' ) led.value(0) else
70             :
71                 imprimir ( 'LED
72 APAGADO' ) led.valor(1) si
73 led.valor() == 1: gpio_state =
74 'APAGADO'         más :
75 gpio_state = 'ENCENDIDO'
76
77
```

¿Que sigue?

¿Que sigue?

¡GRACIAS por participar en esta experiencia de aprendizaje!

Hemos llegado al final de este Tutorial. Si encuentra errores, omisiones o tiene sugerencias y/o preguntas sobre el Tutorial o el contenido de los componentes de este Kit, no dude en contactarnos: support@freenove.com

Haremos todo lo posible para realizar cambios y corregir errores tan pronto como sea posible y publicar una versión revisada.

Si desea obtener más información sobre Arduino, Raspberry Pi, Smart Cars, Robótica y otros productos interesantes en ciencia y tecnología, continúe visitando nuestro sitio web. Continuaremos lanzando productos divertidos, rentables, innovadores y emocionantes.

<https://www.freenove.com/>

Gracias de nuevo por elegir los productos Freenove.

¿Cualquier duda?  support@freenove.com