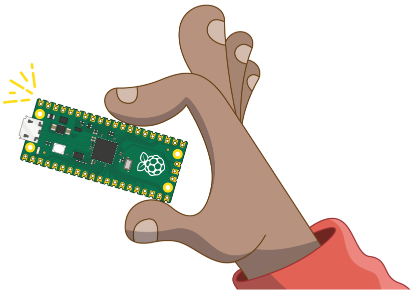
Projects

Getting started with Raspberry Pi Pico

How to start programming you Raspberry Pi Pico with Thonny and MicroPython



Step 1 Introduction

In this project, you will connect a Raspberry Pi Pico to another computer and learn how to program it using MicroPython.

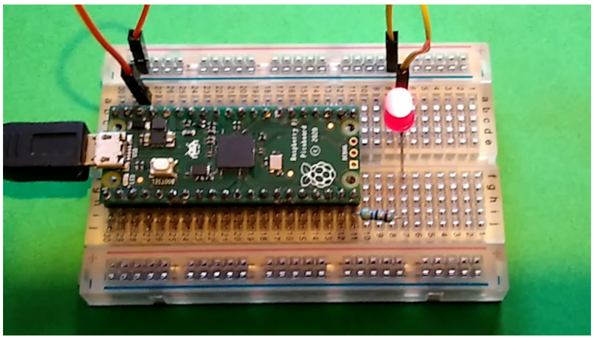
A Raspberry Pi Pico is a low-cost microcontroller device. Microcontrollers are tiny computers, but they tend to lack large volume storage and peripheral devices that you can plug in (for example, keyboards or monitors).

A Raspberry Pi Pico has GPIO pins, much like a Raspberry Pi computer, which means it can be used to control and receive input from a variety of electronic devices.

> **The new Introduction to Raspberry Pi Pico path** (https://projects.raspberrypi.org/en/pathways/pico-intro) uses the **picozero** (https://picozero.readthedocs.io/en/latest/) package to engage in some creative physical computing projects.

What you will make

You will connect a Raspberry Pi Pico to your computer, install the Thonny Python IDE, and write a MicroPython program to blink the onboard LED. If you have additional components available, then you can also try out some more examples.



What you will need

Hardware

- A Raspberry Pi Pico with soldered headers
- A computer that can run the Thonny IDE and program a Raspberry Pi Pico
- A micro USB cable
- A selection of electronics components, such as a button, an LED with appropriate resistor, and a potentiometer (optional)
- A breadboard and M-M jumper leads for connecting additional components (optional)
- An external 5V micro USB power source (optional)

Software

The project will guide you through the installation of:

- MicroPython firmware for Raspberry Pi Pico
- The Thonny Python IDE

What you will learn

- How to load the MicroPython firmware onto a Raspberry Pi Pico
- How to program a Raspberry Pi Pico using MicroPython
- How to connect additional components to a Raspberry Pi Pico and write MicroPython programs to interact with them

Additional information for educators

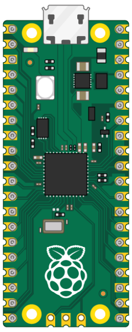
If you are completing this project in a school or other setting with a managed network, then you should make sure that you have the appropriate permissions to mount a USB drive and install software.

If you need to print this project, please use the **printer-friendly version** (https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/print).

Here is a link to the completed scripts for this project (https://rpf.io/p/en/getting-started-with-the-pico-get).

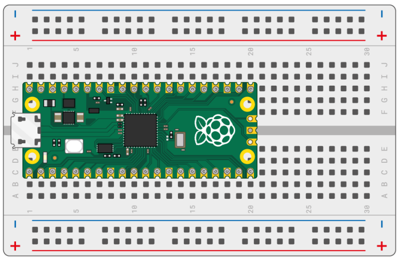
Step 2 Meet Raspberry Pi Pico

This is a Raspberry Pi Pico. Hopefully your device has already had the header pins soldered on, but if not, you might like to have a look at our **Getting started with soldering resource** (https://projects.raspberrypi.org/en/projects/getting-started-with-soldering).

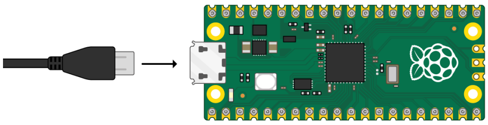


If you have a breadboard, put your Raspberry Pi Pico on the board. ✓

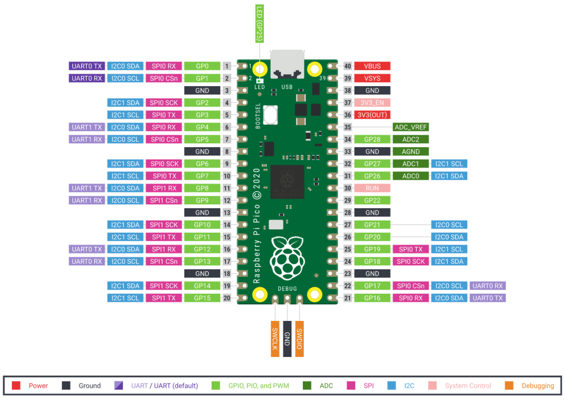
Place it so that the two headers are separated by the ravine in the middle.



Plug your micro USB cable into the port on the left-hand side of the board. ✓



If you need to know the pin numbers for a Raspberry Pi Pico, you can refer to the following diagram.


LED (GP25)
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020
DEBUG
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UART0 RX
21 GP16 SPI0 RX I2C0 SDA UART0 TX
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging

Step 3 Install Thonny

In this step, you will install Thonny or make sure you have the latest version. Then you will connect to a Raspberry Pi Pico and run some simple Python code using the Shell.

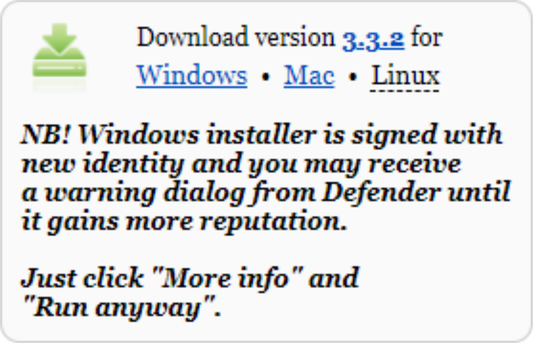
Thonny on Raspberry Pi

- Thonny is already installed on Raspberry Pi OS, but may need to be updated to the latest version
- Open a terminal window, either by clicking the icon in the top left-hand corner of the screen or by pressing the Ctrl+Alt+T keys at the same time
- In the window, type the following to update your OS and Thonny

```
sudo apt update && sudo apt upgrade -y
```

Install Thonny on other operating systems

- On Windows, macOS, and Linux, you can install the latest Thonny IDE or update an existing version
- In a web browser, navigate to **thonny.org** (https://thonny.org/)
- In the top right-hand corner of the browser window, you will see download links for Windows and macOS, and instructions for Linux
- Download the relevant files and run them to install Thonny

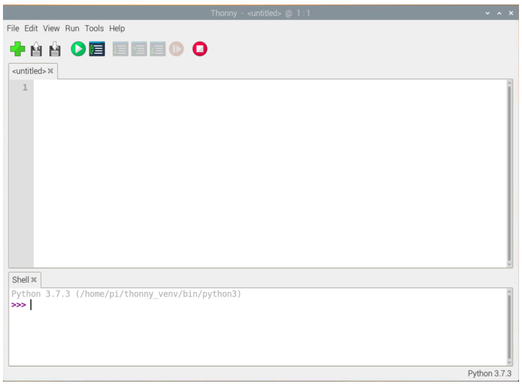


Download version 3.3.2 for
Windows • Mac • Linux

NB! Windows installer is signed with new identity and you may receive a warning dialog from Defender until it gains more reputation.

Just click "More info" and "Run anyway".

Open Thonny from your application launcher. It should look something like this:



```
Python 3.7.3 (/home/pi/thonny_venv/bin/python3)
>>> |
```

You can use Thonny to write standard Python code. Type the following in the main window, and then click the **Run** button (you will be asked to save the file). ☑

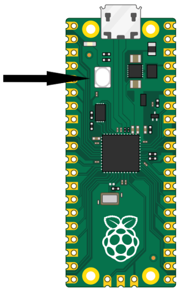
```
print('Hello World!')
```

You’re now ready to move on to the next step and connect your Raspberry Pi Pico.

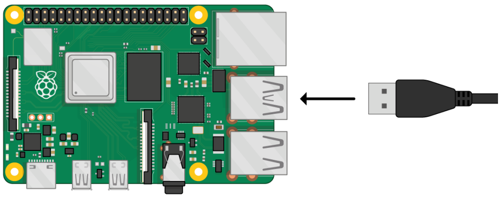
Step 4 Add the MicroPython firmware

If you have never used MicroPython on your Raspberry Pi Pico, you will need to add the MicroPython firmware.

Find the BOOTSEL button on your Raspberry Pi Pico. ☑

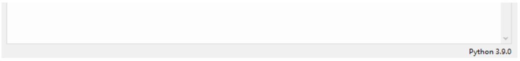


Press the BOOTSEL button and hold it while you connect the other end of the micro USB cable to your computer. A Raspberry Pi is shown in the image below, but the same applies to any computer.

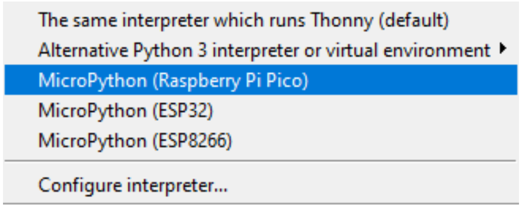


This puts your Raspberry Pi Pico into USB mass storage device mode.

In the bottom right-hand corner of the Thonny window, you will see the version of Python that you are currently using. ☑



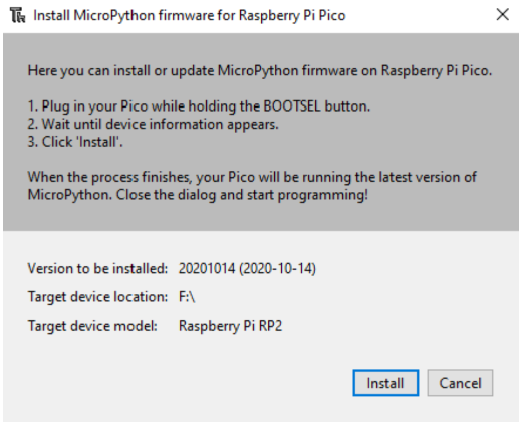
Click on the Python version and choose ‘MicroPython (Raspberry Pi Pico)’:


The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment ▸
MicroPython (Raspberry Pi Pico)
MicroPython (ESP32)
MicroPython (ESP8266)
Configure interpreter...

If you don’t see this option, then check that you have plugged in your Raspberry Pi Pico.

A dialog box will pop up to install the latest version of the MicroPython firmware on your Raspberry Pi Pico.

Click the **Install** button to copy the firmware to your Raspberry Pi Pico.



Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: 20201014 (2020-10-14)

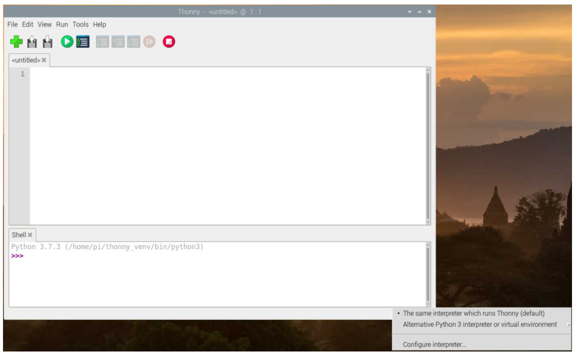
Target device location: F:\

Target device model: Raspberry Pi RP2

Wait for the installation to complete and click **Close**.

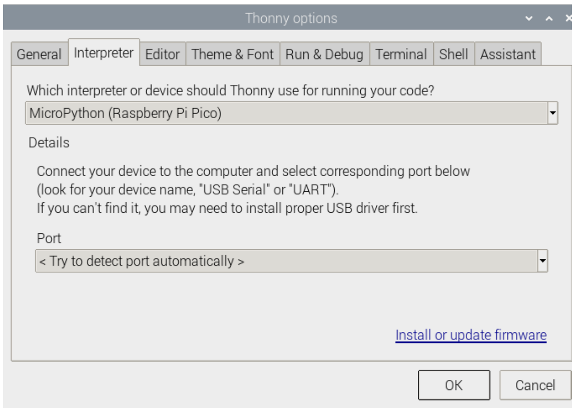
Firmware installation menu

You can also access the firmware installation menu if you click on ‘MicroPython (Raspberry Pi Pico)’ in the status bar and choose ‘Configure interpreter ...’.


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Shell
The same interpreter which runs Thonny (default)
Alternative Python 3 interpreter or virtual environment
Configure interpreter...

```
Python 3.7.3 (/home/pi/thonny_venv/bin/python3)
>>>
```

The interpreter settings will open.



Which interpreter or device should Thonny use for running your code?

MicroPython (Raspberry Pi Pico)

Details

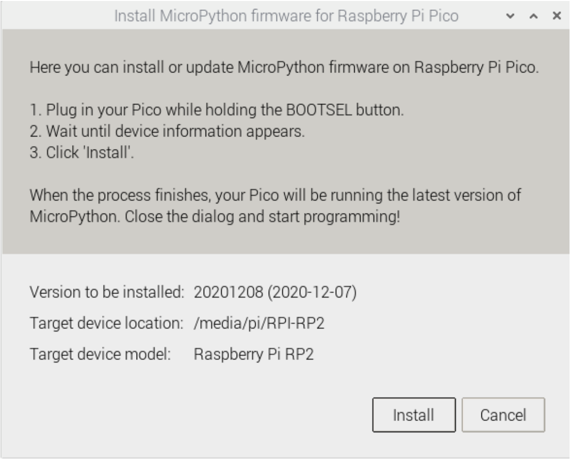
Connect your device to the computer and select corresponding port below
(look for your device name, "USB Serial" or "UART").
If you can't find it, you may need to install proper USB driver first.

Port

< Try to detect port automatically >

Click on **Install or update firmware**.

You will be prompted to plug in your Raspberry Pi Pico while you hold the BOOTSEL button.



Install MicroPython firmware for Raspberry Pi Pico

Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

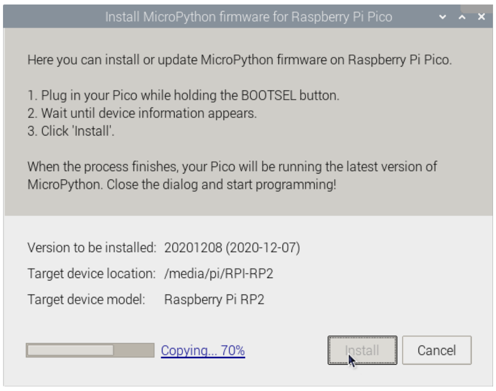
When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: 20201208 (2020-12-07)

Target device location: /media/pi/RPI-RP2

Target device model: Raspberry Pi RP2

Then you can click **Install**.



Install MicroPython firmware for Raspberry Pi Pico

Here you can install or update MicroPython firmware on Raspberry Pi Pico.

1. Plug in your Pico while holding the BOOTSEL button.
2. Wait until device information appears.
3. Click 'Install'.

When the process finishes, your Pico will be running the latest version of MicroPython. Close the dialog and start programming!

Version to be installed: 20201208 (2020-12-07)

Target device location: /media/pi/RPI-RP2

Target device model: Raspberry Pi RP2

Copying... 70%

Wait for the installation to complete and click **Close**.

You don’t need to update the firmware every time you use your Raspberry Pi Pico. Next time, you can just plug it into your computer without pressing the BOOTSEL button.

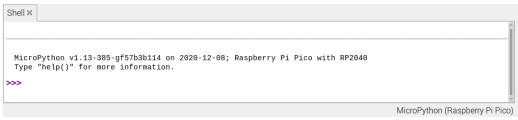
Step 5 Use the Shell

In this step, you will use the Thonny Shell to run some simple Python code on your Raspberry Pi Pico.

Make sure that your Raspberry Pi Pico is connected to your computer and you have selected the MicroPython (Raspberry Pi Pico) interpreter.

Look at the Shell panel at the bottom of the Thonny editor.

You should see something like this:



```
 MicroPython v1.13-385-gf57b3b114 on 2020-12-08; Raspberry Pi Pico with RP2040
 Type "help()" for more information.
>>>
```

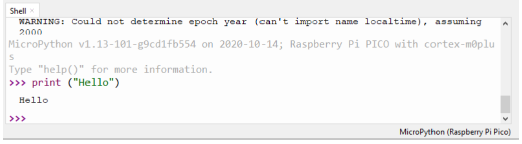
Thonny is now able to communicate with your Raspberry Pi Pico using the REPL (read–eval–print loop), which allows you to type Python code into the Shell and see the output.

Now you can type commands directly into the Shell and they will run on your Raspberry Pi Pico.

Type the following command.

```
print("Hello")
```

Tap the Enter key and you will see the output:



```
  WARNING: Could not determine epoch year (can't import name localtime), assuming
  2000
MicroPython v1.13-101-g9cd1fb554 on 2020-10-14; Raspberry Pi PICO with cortex-m0plu
s
Type "help()" for more information.
>>> print ("Hello")
  Hello
>>>
```

MicroPython adds hardware-specific modules, such as `machine`, that you can use to program your Raspberry Pi Pico.

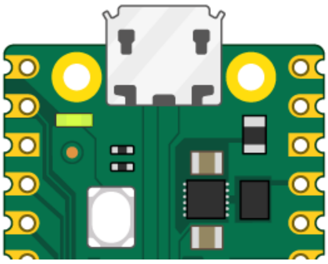
Let's create a `machine.Pin` object to correspond with the onboard LED, which can be accessed using GPIO pin 25.

If you set the value of the LED to `1`, it turns on.

Enter the following code, make sure you tap Enter after each line.

```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```

You should see the onboard LED light up.



Type the code to set the value to `0` to turn the LED off.

```
led.value(0)
```

Turn the LED on and off as many times as you like.

Tip: You can use the up arrow on the keyboard to quickly access previous lines.

If you want to write a longer program, then it's best to save it in a file. You'll do this in the next step.

Step 6 Blink the onboard LED

The Shell is useful to make sure everything is working and try out quick commands. However, it's better to put longer programs in a file.

Thonny can save and run MicroPython programs directly on your Raspberry Pi Pico.

In this step, you will create a MicroPython program to blink the onboard LED on and off in a loop.

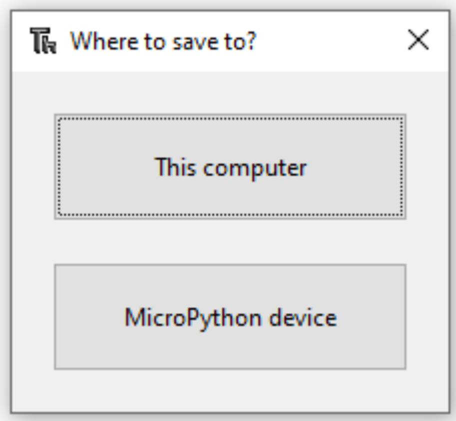
Click in the main editor pane of Thonny. ✓

Enter the following code to toggle the LED.

```
from machine import Pin
led = Pin(25, Pin.OUT)

led.toggle()
```

Click the **Run** button to run your code. ✓

Thonny will ask whether you want to save the file on **This computer** or the **MicroPython device**. Choose **MicroPython device**.



Enter `blink.py` as the file name.

Tip: You need to enter the `.py` file extension so that Thonny recognises the file as a Python file.

Thonny can save your program to your Raspberry Pi Pico and run it.

You should see the onboard LED switch between on and off each time you click the **Run** button.

You can use the `Timer` module to set a timer that runs a function at regular intervals.

Update your code so it looks like this:

```
from machine import Pin, Timer
led = Pin(25, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

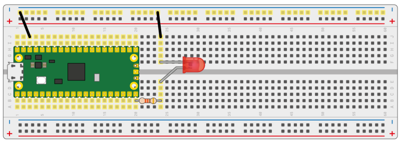
Click **Run** and your program will blink the LED on and off until you click the **Stop** button.

Save your project

Step 7 Use digital inputs and outputs

Now you know the basics, you can learn to control an external LED with your Raspberry Pi Pico, and get it to read input from a button.

Use a resistor between about 50 and 330 ohms, an LED, and a pair of M-M jumper leads to connect up your Raspberry Pi Pico as shown in the image below. ☑



In this example, the LED is connected to pin 15. If you use a different pin, remember to look up the number in the pinout diagram in the **Meet Raspberry Pi Pico section** (1.html).

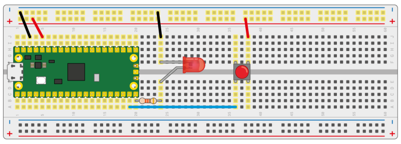
Use the same code as you did to blink the onboard LED, but change the pin number to 15. ☑

```
from machine import Pin, Timer
led = Pin(15, Pin.OUT)
timer = Timer()

def blink(timer):
    led.toggle()

timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

Run your program and your LED should start to blink. If it's not working, check your wiring to be sure that the LED is connected.

Next, let's try and control the LED using a button.

Add a button to your circuit as shown in the diagram below. ✓



The button is on pin `14`, and is connected to the 3.3V pin on your Raspberry Pi Pico. This means when you set up the pin, you need to tell MicroPython that it is an input pin and needs to be *pulled down*.

Create a new file and add this code. ✓

```
from machine import Pin
import time

led = Pin(15, Pin.OUT)
button = Pin(14, Pin.IN, Pin.PULL_DOWN)

while True:
    if button.value():
        led.toggle()
        time.sleep(0.5)
```

Run your code and then when you press the button, the LED should toggle on or off. If you hold the button down, it will flash. ✓

Save your project

Step 8 Control LED brightness with PWM

Pulse width modulation (https://en.wikipedia.org/wiki/Pulse-width_modulation), allows you to give analogue behaviours to digital devices, such as LEDs. This means that rather than an LED being simply on or off, you can control its brightness.

For this activity, you can use the circuit from the last step.

Open a new file in Thonny and add the following code. ☑

```
from machine import Pin, PWM
from time import sleep

pwm = PWM(Pin(15))

pwm.freq(1000)

while True:
    for duty in range(65025):
        pwm.duty_u16(duty)
        sleep(0.0001)
    for duty in range(65025, 0, -1):
        pwm.duty_u16(duty)
        sleep(0.0001)
```

Save and run the file. You should see the LED pulse bright and dim, in a continuous cycle. ☑

The **frequency** (`pwm.freq`) tells Raspberry Pi Pico how often to switch the power between on and off for the LED.

The duty cycle tells the LED for how long it should be on each time. For Raspberry Pi Pico in MicroPython, this can range from `0` to `65025`. `65025` would be 100% of the time, so the LED would stay bright. A value of around `32512` would indicate that it should be on for half the time.

Have a play with the `pwm.freq()` values and the `pwm.duty_u16` values, as well as the length of time for the `sleep`, to get a feel for how you can adjust the brightness and pace of the pulsing LED. ☑

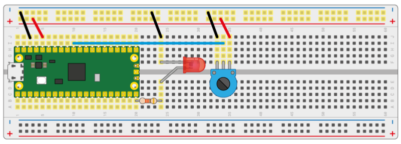
Save your project

Step 9 Control an LED with an analogue input

Your Raspberry Pi Pico has input pins that can receive analogue signals. This means that instead of only reading the values of 1 and 0 (on and off), it can read values in between.

A potentiometer is the perfect analogue device for this activity.

Replace the button in your circuit with a potentiometer. Follow the wiring diagram below to connect it to an analogue pin. ☑



In a new file in Thonny, you can first read the resistance of the potentiometer. ☑

Add this code to a new file, and then run it.

```
from machine import ADC, Pin
import time

adc = ADC(Pin(26))

while True:
    print(adc.read_u16())
    time.sleep(1)
```

Turn the potentiometer to see your maximum and minimum values.

They should be approximately between 0 and 65025.

You can now use this value to control the duty cycle for PWM on the LED.

Change the code to the following. Once you have run it, tune the dial on the potentiometer to control the LED’s brightness.

```
from machine import Pin, PWM, ADC

pwm = PWM(Pin(15))
adc = ADC(Pin(26))

pwm.freq(1000)

while True:
    duty = adc.read_u16()
    pwm.duty_u16(duty)
```

Save your project

Step 10 Power your Raspberry Pi Pico

If you want to run your Raspberry Pi Pico without it being attached to a computer, you need to use a USB power supply.

Safe operating voltages are between 1.8V and 5.5V.

To automatically run a MicroPython program, simply save it to the device with the name `main.py`

In Thonny, click on the **File** menu and then **Save as** for the last program you wrote. ☑

When prompted, select 'MicroPython device' from the pop-up menu. ☑

Name your file `main.py` ☑

You can now disconnect your Raspberry Pi Pico from your computer and use a micro USB cable to connect it to a mobile power source, such as a battery pack. ☑

Once connected, the `main.py` file should run automatically so you can interact with the components attached to your Raspberry Pi Pico.

Save your project

Step 11 What next?

- Try the new **Introduction to Raspberry Pi Pico path** (https://projects.raspberrypi.org/en/pathways/pico-intro), using the picozero package, to control LEDs and buzzers and read signals from switches and dials.
- Why not try out a few more components with your Raspberry Pi Pico — perhaps a buzzer, a light-dependent resistor (LDR), or even a motor controller
- For further guidance on using your Raspberry Pi Pico, you can have a look at the **documentation here** (https://www.raspberrypi.org/documentation/pico/getting-started/)

Did you enjoy the project? Have you spotted a mistake? Please click the **Send feedback** button below and let us know!