# ENGR 3421: Robotics I

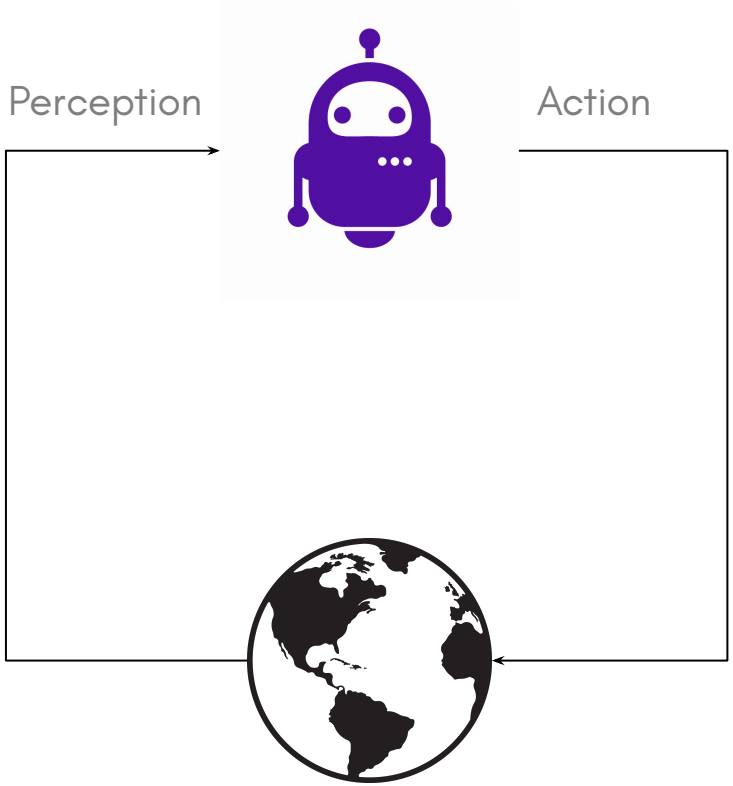
## Raspberry Pi Pico

09/02/2025

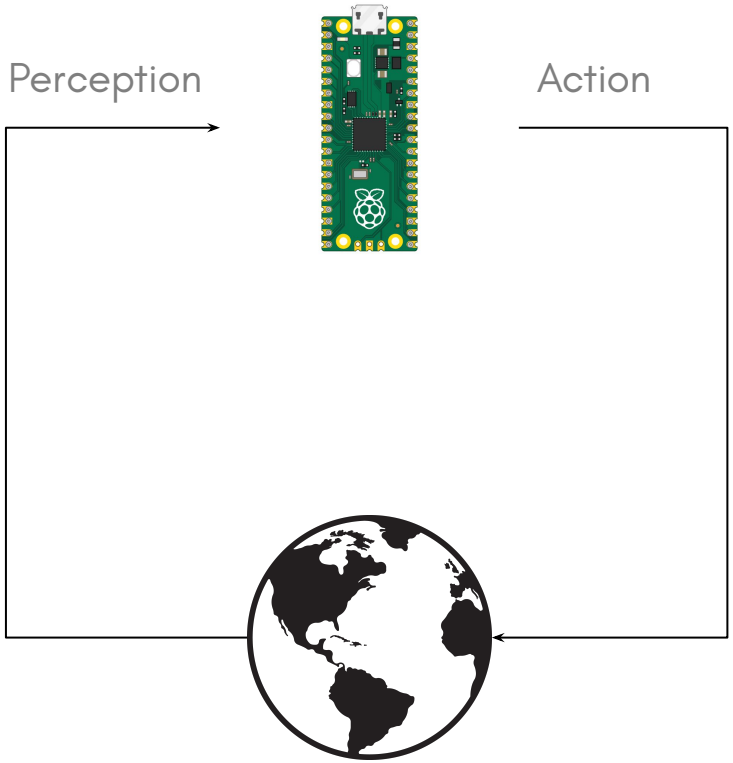
# Outline

- Introduction to Raspberry Pi Pico
- MicroPython
- GPIO

# A Robot Needs A Brain

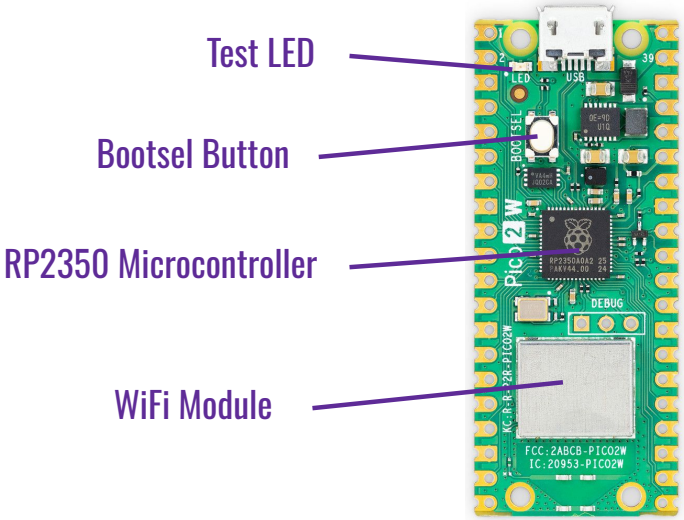


# A Robot Needs A Brain



# Overview

Raspberry Pi Pico 2 W is a microcontroller development board made by Raspberry Pi Foundation.



# Features

- Dual Cortex-M33 or RISC-V Hazard3 cores clocked at up to 150MHz
- 520 kB multi-bank high performance SRAM
- 4MB of on-board Flash memory
- On-board single-band 2.4GHz wireless interfaces (802.11n, Bluetooth 5.2)
- Micro USB B port for power and data (USB 1.1)
- Exposes 26 multi-function 3.3V general purpose I/O (GPIO)
- 2 × UART, 2 × I2C, 2 × SPI, 24 × PWM channels, 1× HSTX peripheral
- 1 × timer with 4 alarms, 1 × AON Timer

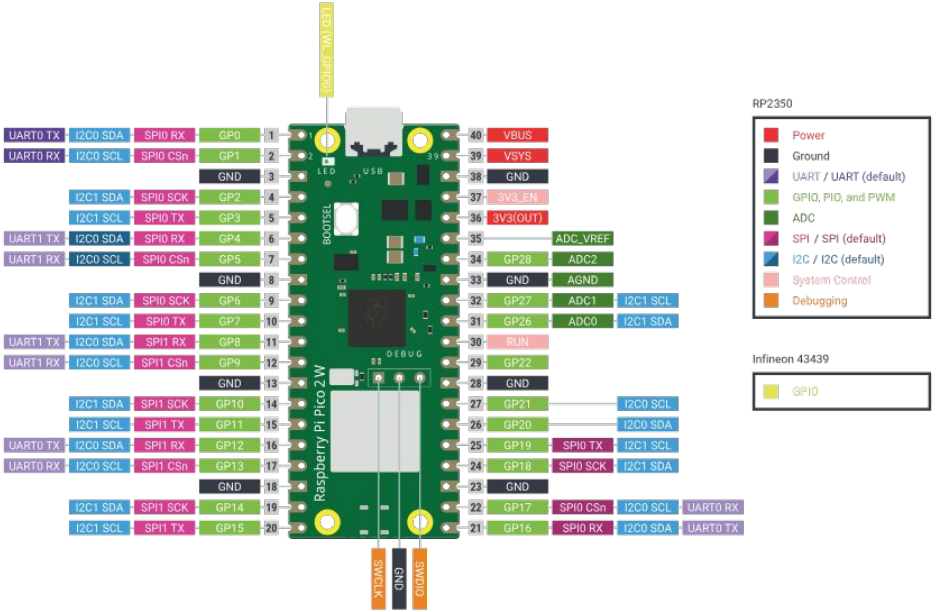
# Pico Projects

- LCD Display
- PicoLight (LED control)
- Matrix Touch Keypad
- Zapper Gun (game controller)
- Music Box
- Wood Burning Plotter
- Pico SMARS (mobile robot)

# Pinout – Power Pins

Pico uses an on-board buck-boost SMPS which is able to generate the required 3.3V from a wide range of input voltages (~1.8 to 5.5V). This allows significant flexibility in powering the unit from various sources such as a single Lithium-Ion cell, or 3 AA cells in series.

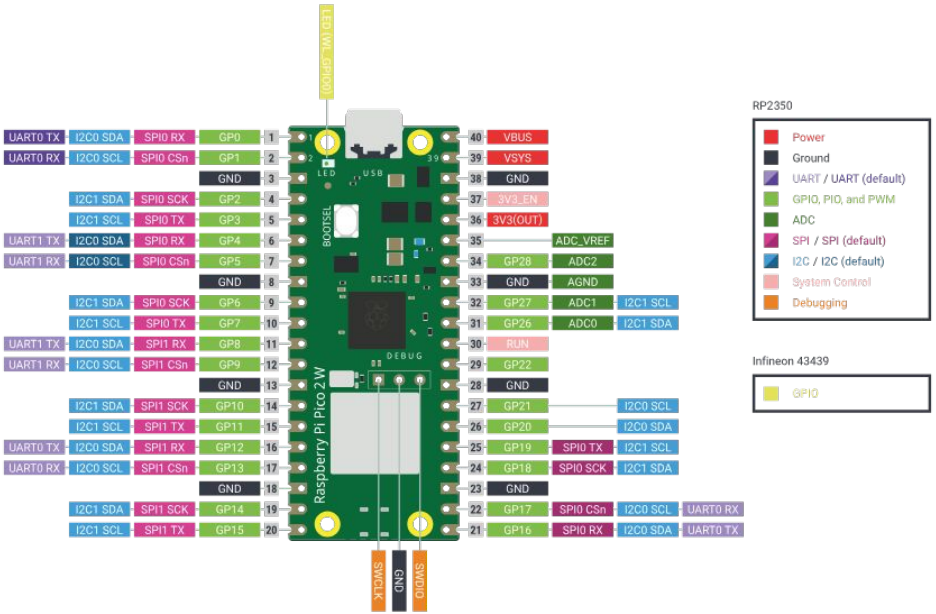
- **VBUS(OUT) –** connected to micro-USB port pin 1. This is nominally 5V (or 0V if the USB is not connected or not powered).
- **VSYS(IN)** – main system input voltage, which can vary in the allowed range 1.8V to 5.5V, and is used by the on-board SMPS to generate the 3.3V for the RP2040 and its GPIO.
- **3V3(OUT)** – This is a 3.3V output, from the Pico's internal regulator. It can be used to power additional components, providing you keep the load under 300ma.



# Pinout – Ground Pins

There are 9 ground pins in total.

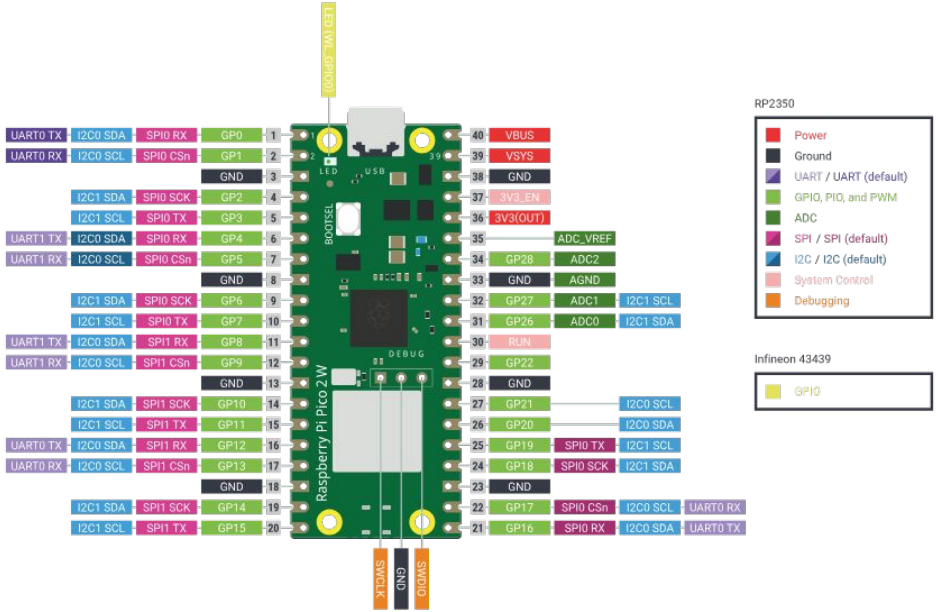
- Evenly spaced.
- Square pads.



# Pinout – GPIO Pins

There are 26 multi-function GPIO pins.

- They can be programmed to receive or send signals.
- “WL_GPIO0” is connect to the on-board LED.
- GP26, GP27, GP28 can be configured as ADC.
- 2×SPI, 2×I2C, 2×UART

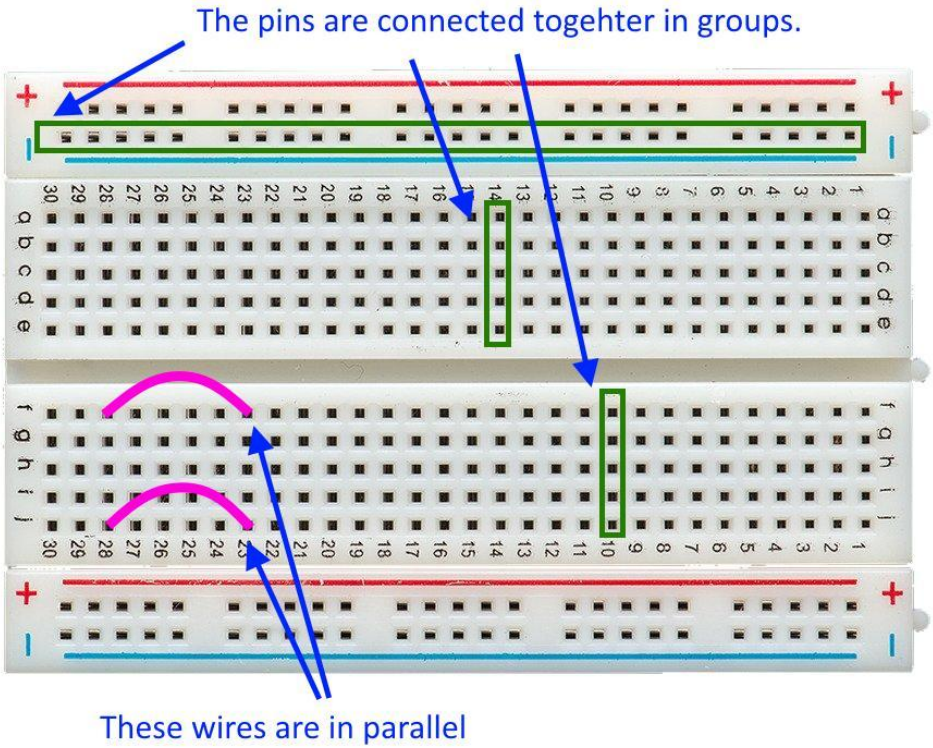


# Pins Notes

- DO NOT connect a power pin to any GND pins directly without protection from a resistor.
- GPIO pins use 3.3V logic for input/output. Do not input 5V signals to Pico.
- 3V3(OUT) can be used to power external circuitry (maximum output current: 300mA).
- A motor will unlikely be driven by GPIO pins output directly.

# Get Started

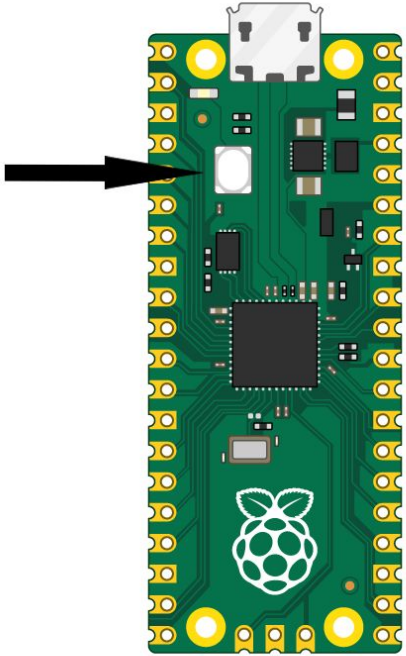
# Solderless Breadboard



# Install Thonny

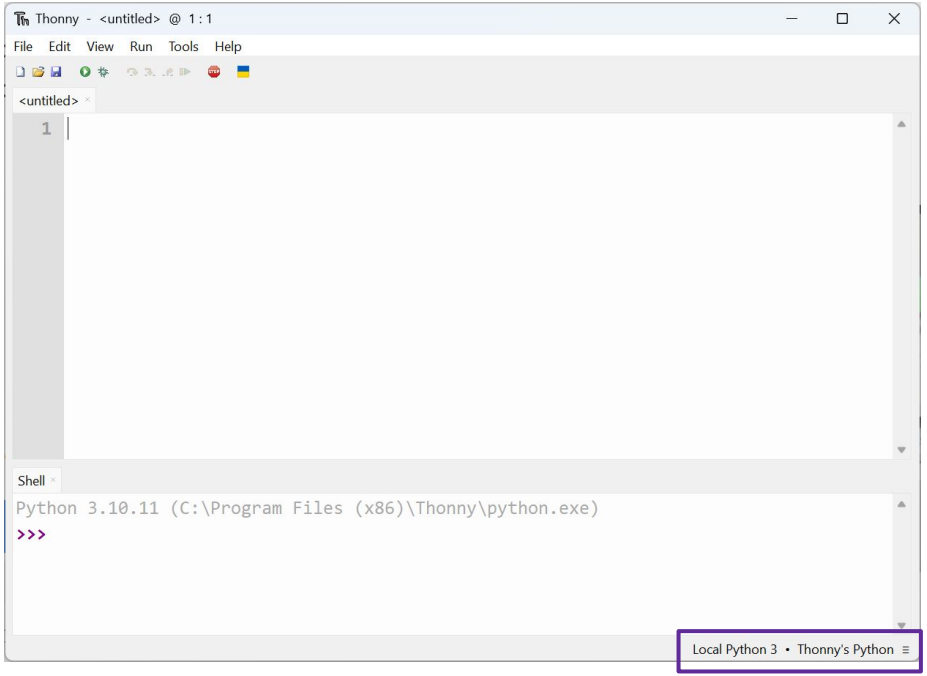

thonny.org

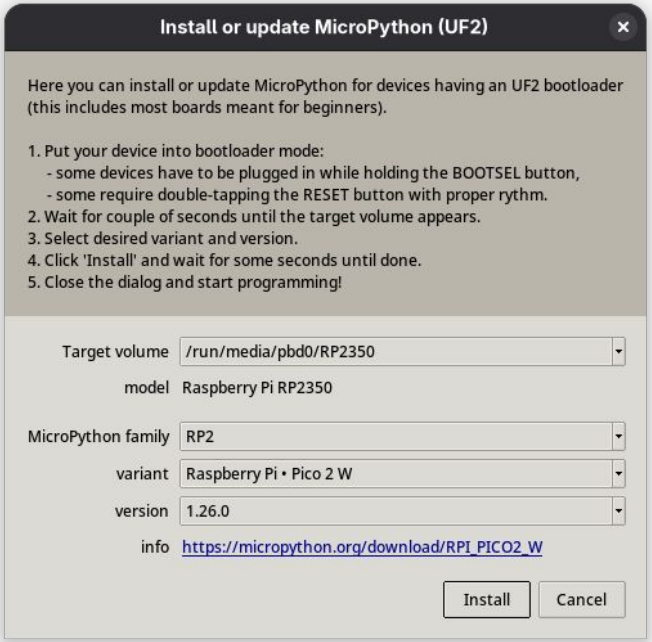
# Entering Bootsel Mode



Press the BOOTSEL button and hold it while you connect the other end of the micro USB cable to your computer.

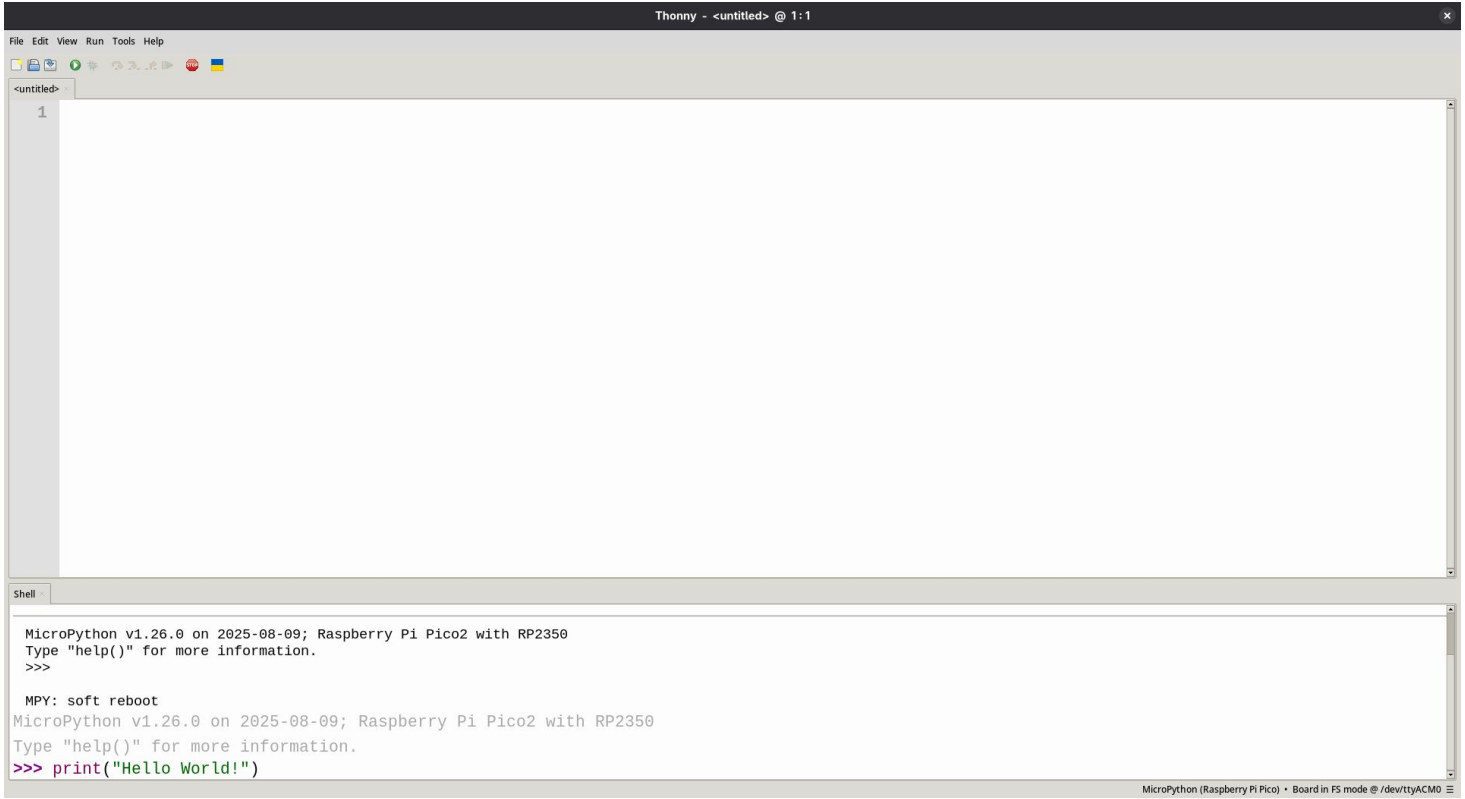
# Install MicroPython




Select “Install MicroPython...”

# Hello World on Pico



# Test code in Shell

- Select right Python interpreter.
- Type after >>>

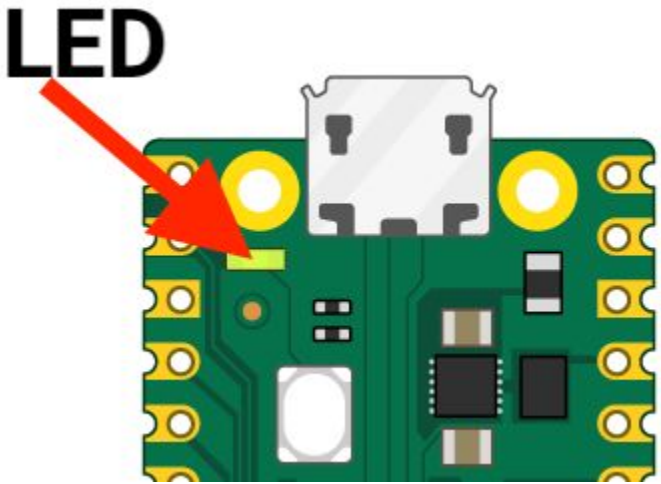
Shell

```
 MPY: soft reboot
MicroPython v1.26.0 on 2025-08-09; Raspberry Pi Pico2 with RP2350
Type "help()" for more information.
>>> import machine
>>> led = machine.Pin("WL_GPIO0", Pin.OUT)
```

MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/ttyACM0

# Code On-Board LED in Shell

```
import machine
led = machine.Pin("WL_GPIO0", Pin.OUT)
print(led)
led.on()
led.off()
led.toggle()
```

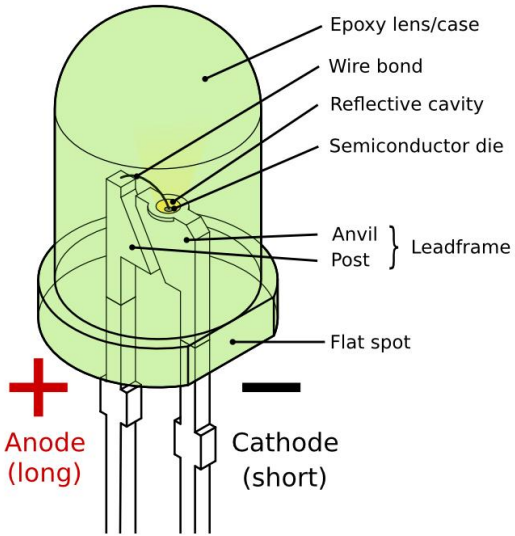


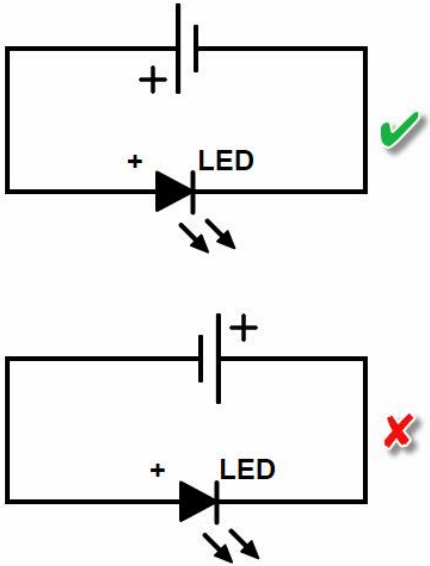

Shell

```
 MPY: soft reboot
MicroPython v1.26.0 on 2025-08-09; Raspberry Pi Pico2 with RP2350
Type "help()" for more information.
>>> import machine
>>> led = machine.Pin("WL_GPIO0", Pin.OUT)
```

MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/ttyACM0

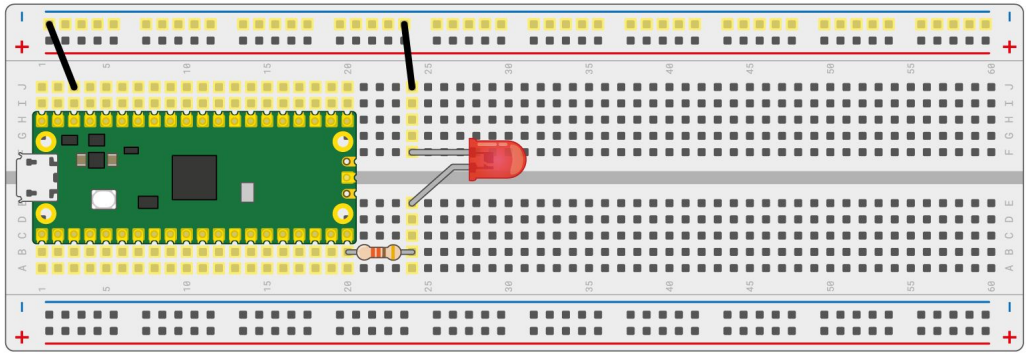
# Light Emitting Diode (LED)



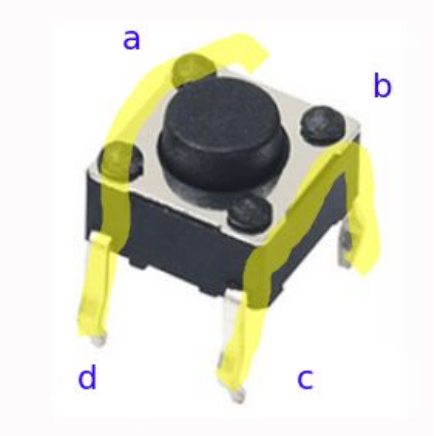



# GPIO Pin Output

```python
import time
import machine

# SETUP
led = machine.Pin(15, machine.Pin.OUT)

# LOOP
while True:
    led.toggle()
    time.sleep(1)
```

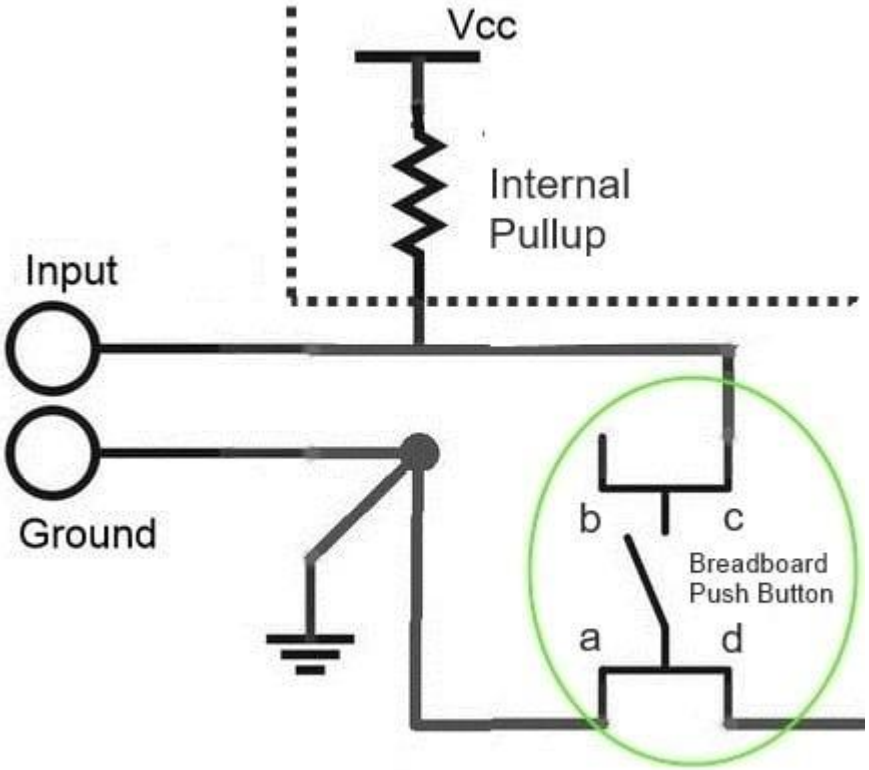


# Switch Button

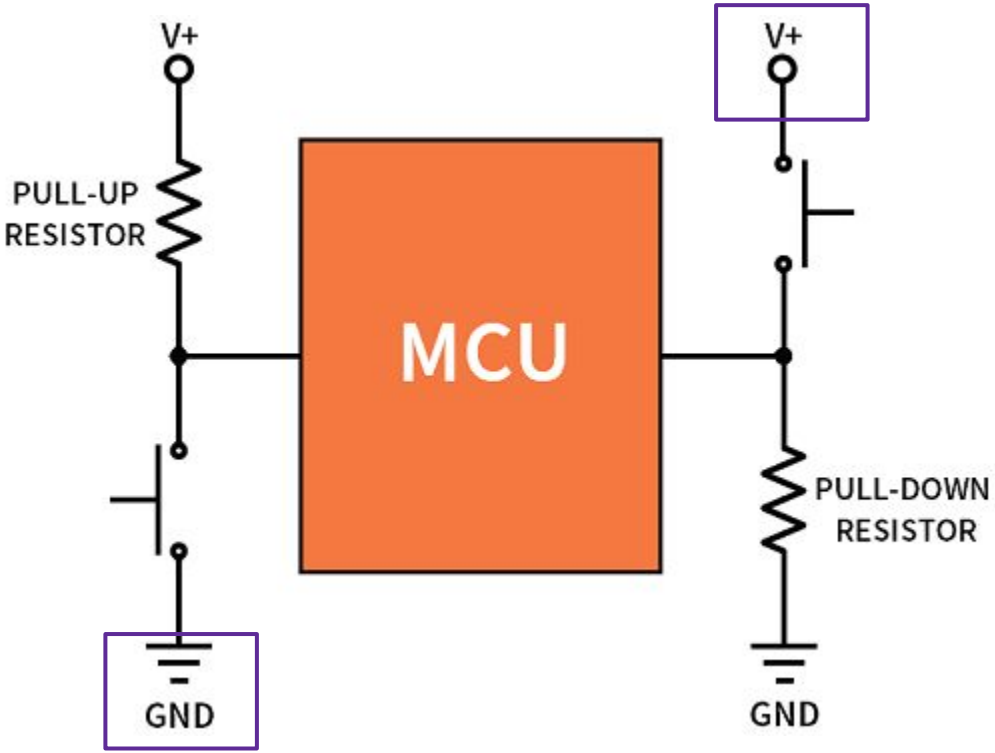



Good combinations:

- ab
- cd
- ac
- bd

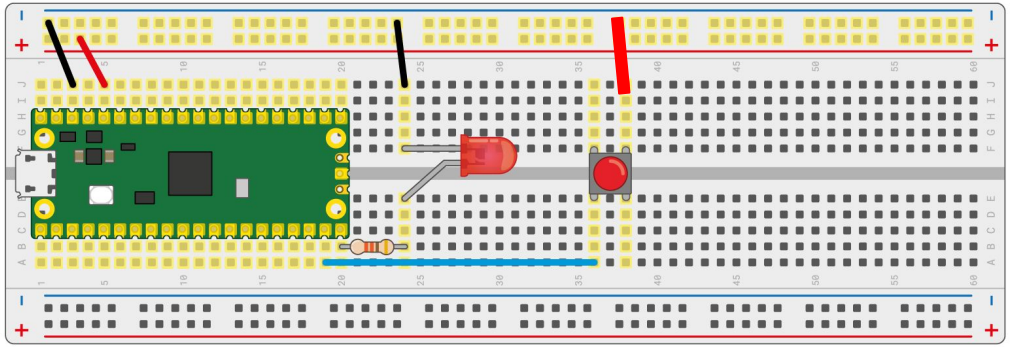


# Pull-Up vs. Pull-Down Resistor



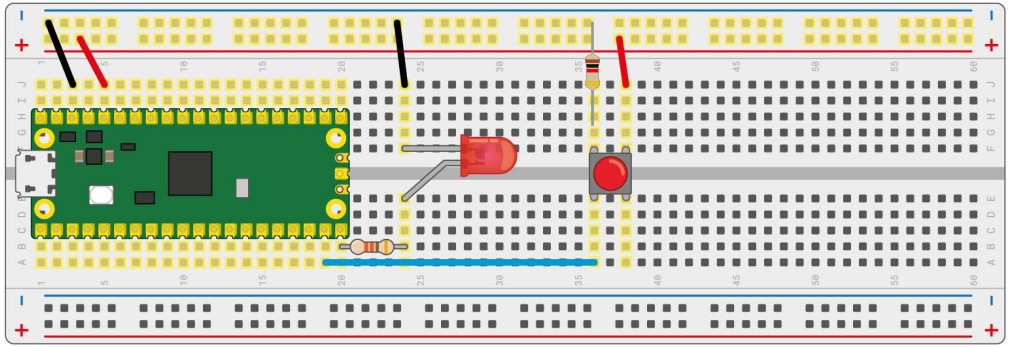
# GPIO Pin Input - PULL_UP

```python
from machine import Pin
from time import sleep

# SETUP
led = Pin(15, Pin.OUT)
button = Pin(14, Pin.IN, Pin.PULL_UP)

# LOOP
while True:
    if button.value() == 0:
        led.toggle()
    sleep(0.1)
```

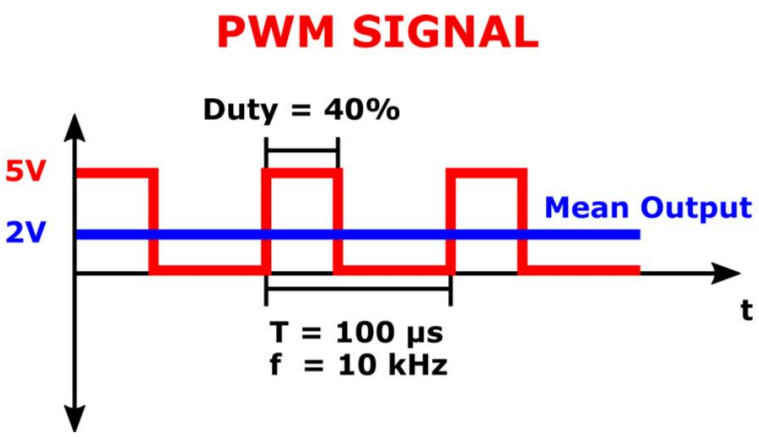


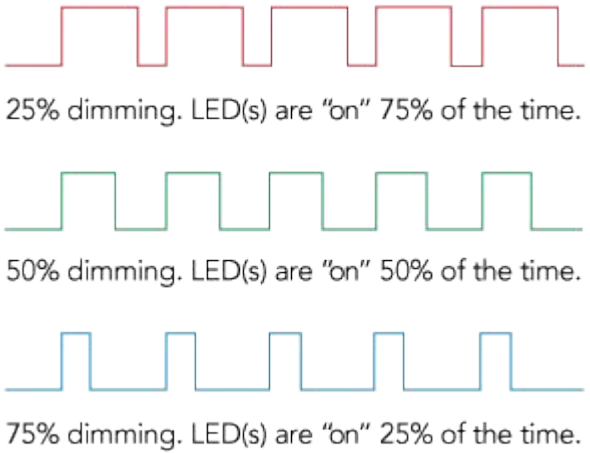
# GPIO Pin Input - PULL_DOWN

```python
from machine import Pin
from time import sleep

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

# LOOP
while True:
    if button.value() == 1:
        led.toggle()
    sleep(0.1)
```



# Pulse Width Modulation (PWM)



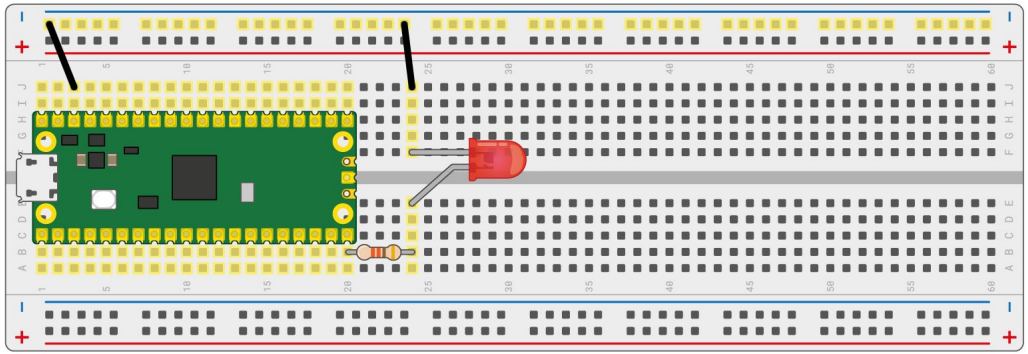




PWM signal properties

- Frequency (f): on/off rate, the higher the more natural
- Duty Cycle: portion of ON, the higher the brighter

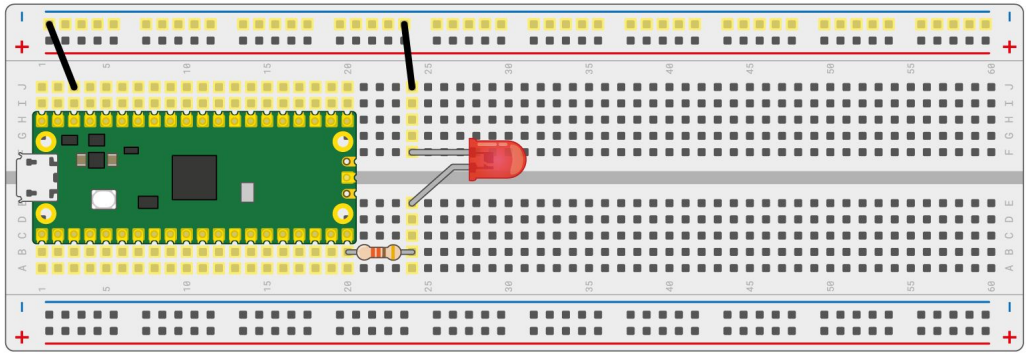
# PWM Controlled LED Brightness

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

# SETUP
dimmer = PWM(Pin(15))
dimmer.freq(1000)

# LOOP
while True:
    for duty in range(65536):
        dimmer.duty_u16(duty)
        sleep(0.0001)
    for duty in range(65535, 0, -1):
        dimmer.duty_u16(duty)
        sleep(0.0001)
```



# Timer

```
from machine import Pin, Timer

# SETUP
led = Pin(15, Pin.OUT)
blink_timer = Timer()

def toggle_led(timer):
    led.toggle()

blink_timer.init(freq=2.5, mode=Timer.PERIODIC, callback=toggle_led)

# LOOP
```



# Interrupt

```python
from machine import Pin

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

def toggle_led(pin):
    led.toggle()

button.irq(trigger=Pin.IRQ_FALLING, handler=toggle_led)

# LOOP
```