ENGR 3421: Robotics I

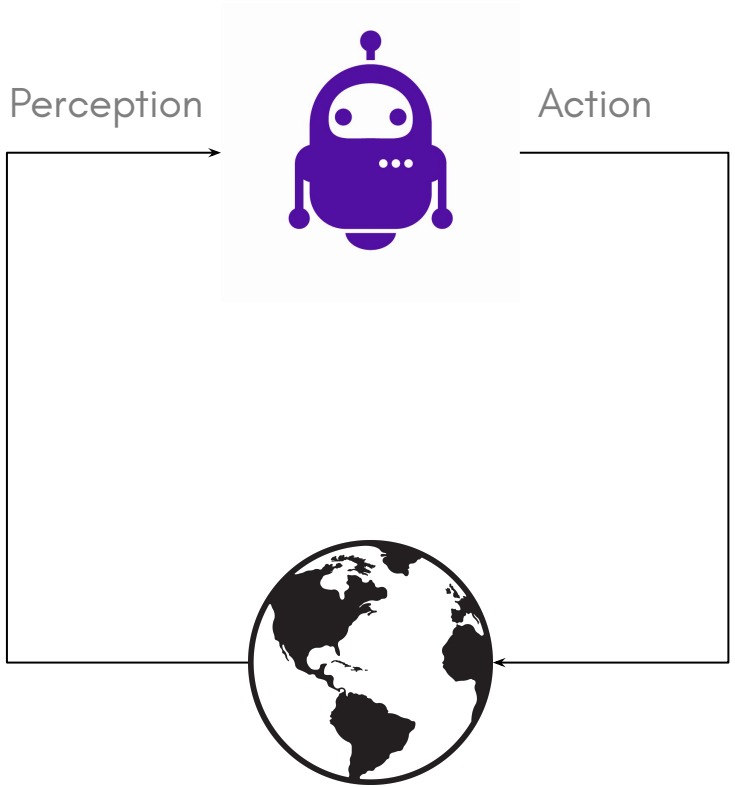
Raspberry Pi Pico

09/03/2024

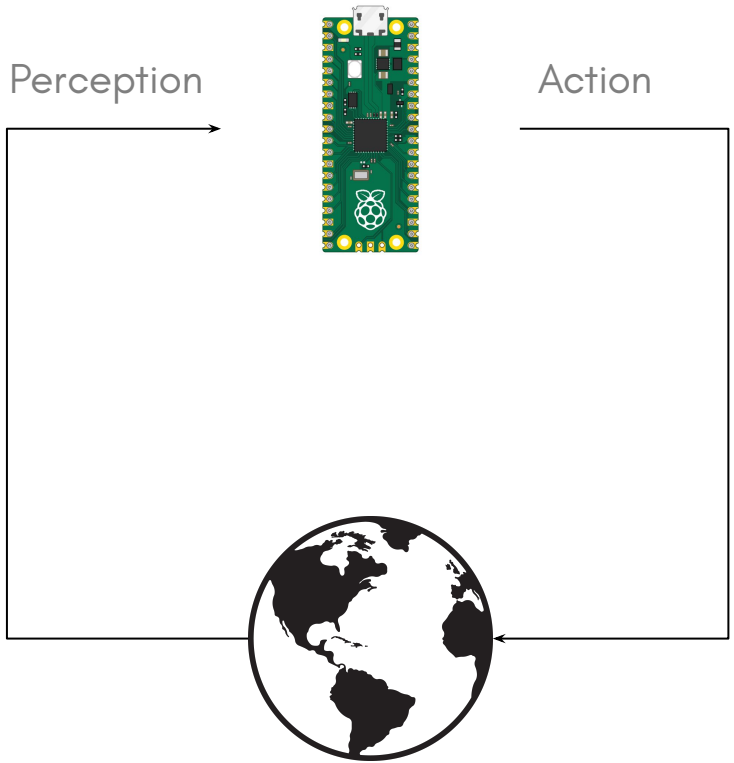
Outline

- Introduction to Raspberry Pi Pico
- MicroPython
- GPIO

A Robot Needs A Brain

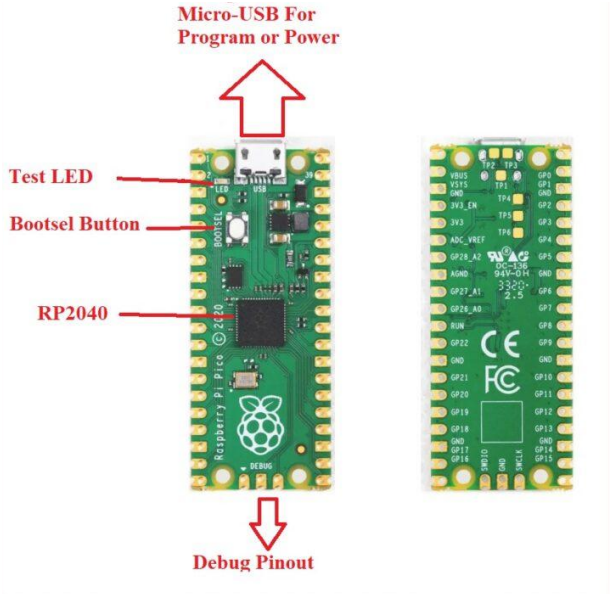


A Robot Needs A Brain



Overview

Raspberry Pi Pico is a microcontroller made by Raspberry Pi Foundation. It is featured with an RP2040 processor based on the ARM Dual-core Cortex architecture.



Features

- Dual-core ARM Cortex M0+ processor, flexible clock running up to 133 MHz
- 264kB of SRAM, and 2MB of on-board Flash memory
- Castellated module allows soldering direct to carrier boards
- USB 1.1 Host and Device support
- Low-power sleep and dormant modes
- Drag & drop programming using mass storage over USB
- 26 multi-function GPIO pins
- 2×SPI, 2×I2C, 2×UART, 3×12-bit ADC, 16×controllable PWM channels
- Accurate clock and timer on-chip
- Temperature sensor
- Accelerated floating point libraries on-chip
- 8×Programmable IO (PIO) state machines for custom peripheral support

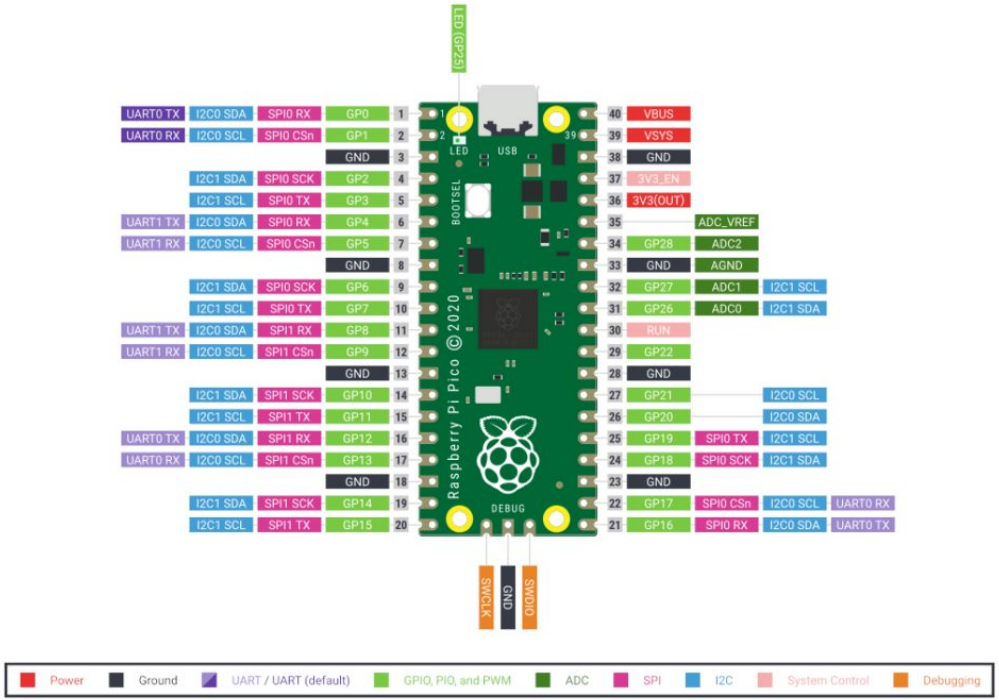
Pico Projects

- 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 (to power RP2040 and external circuitry) 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.

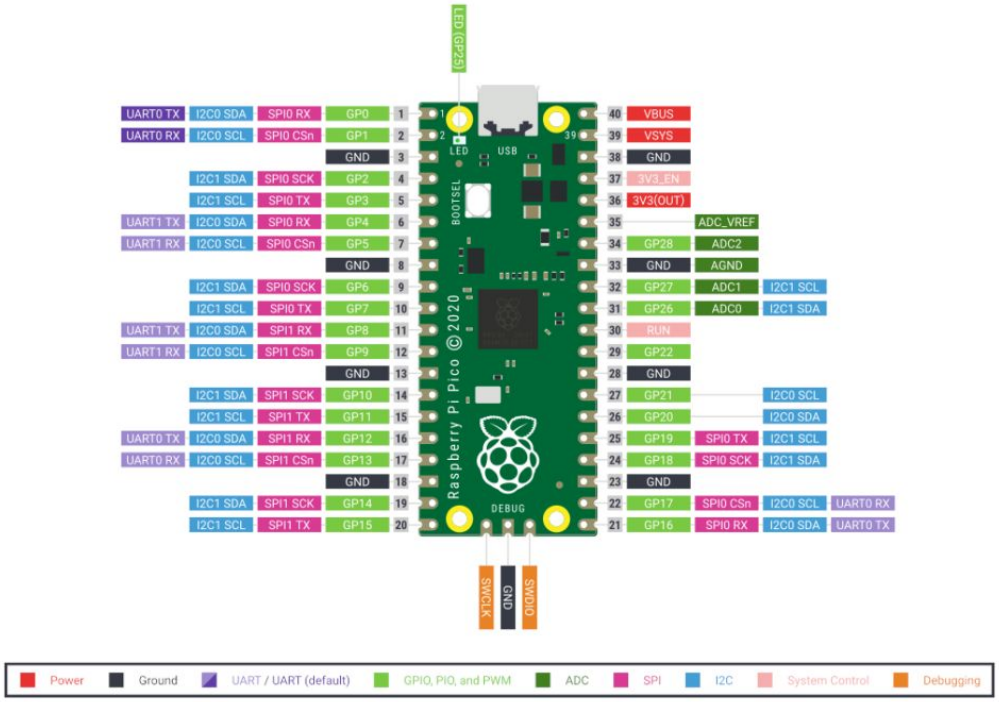
- **VBUS(OUT) –** micro-USB input voltage, connected to micro-USB port pin 1. This is nominally 5V (or 0V if the USB is not connected or not powered).
- **VSYS(IN)** – main system input voltage, which can vary in the allowed range 1.8V to 5.5V, and is used by the on-board SMPS to generate the 3.3V for the RP2040 and its GPIO.
- **3V3(OUT)** – This is a 3.3-volt 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.

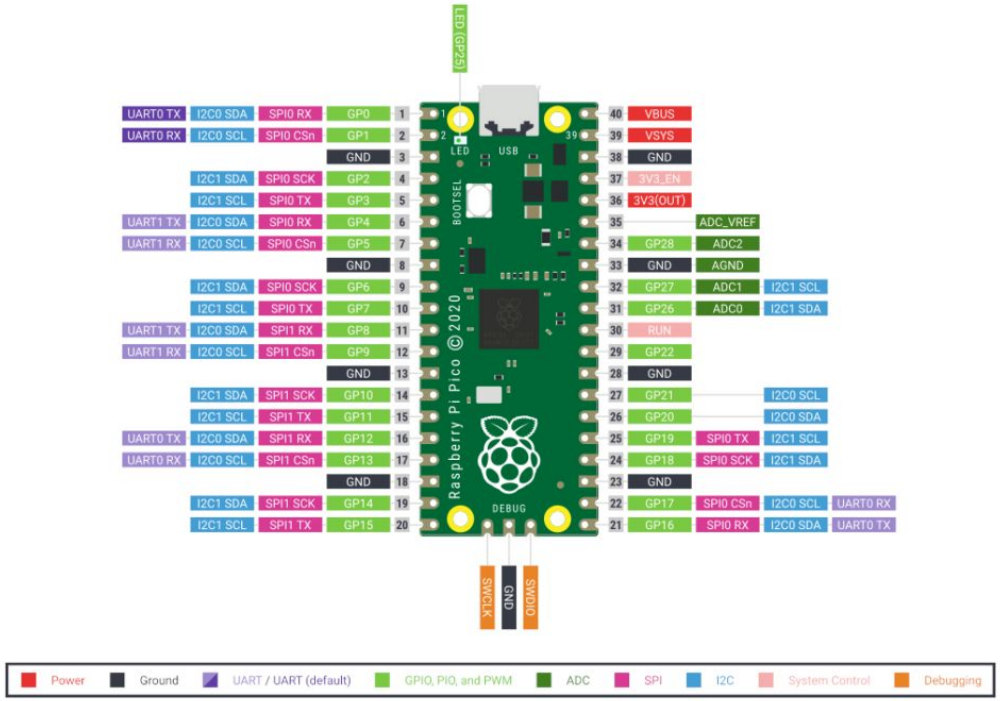
- Evenly spaced.
- Square pads.



Pinout – GPIO Pins

There are 26 multi-function GPIO pins.

- They can be programmed to receive or send signals.
- GP25 is connect to the on-board LED.
- Up to 16 GPIO pins can be configured as PWM.
- GP26, GP27, GP28 can be configured as ADC.
- 2×SPI, 2×I2C, 2×UART

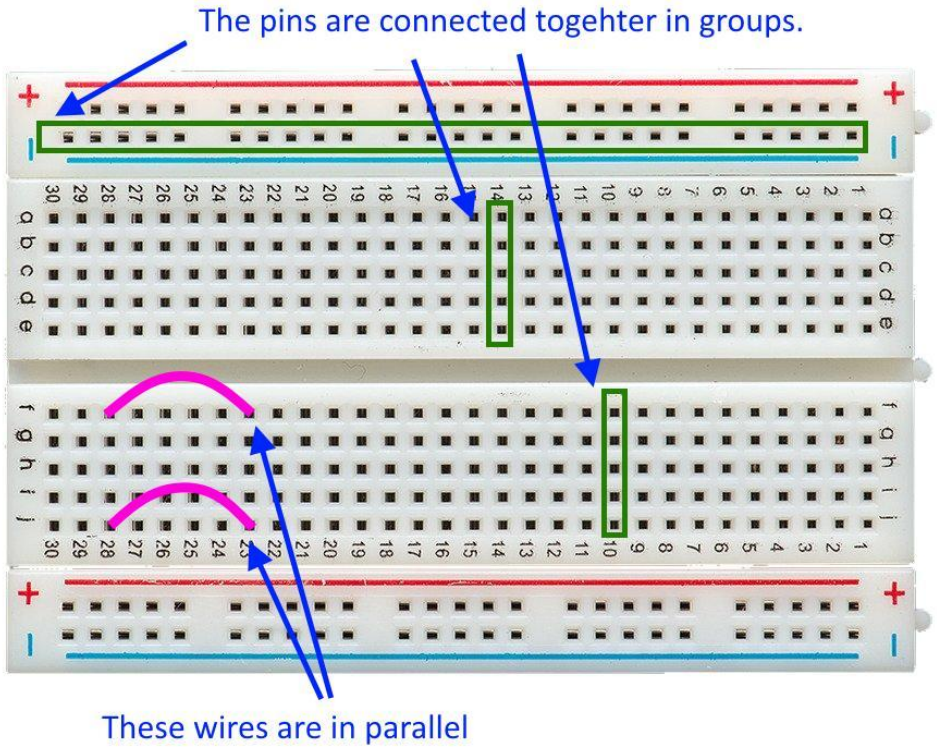


Pins Notes

- DO NOT short connect pins.
- GPIO pins use 3.3V logic for input/output. Do not input 5V signals to GPIO pins.
- Max current draw is ~50mA. Don’t try to drive your motor with GPIO pins directly.

Get Started with Micro Python

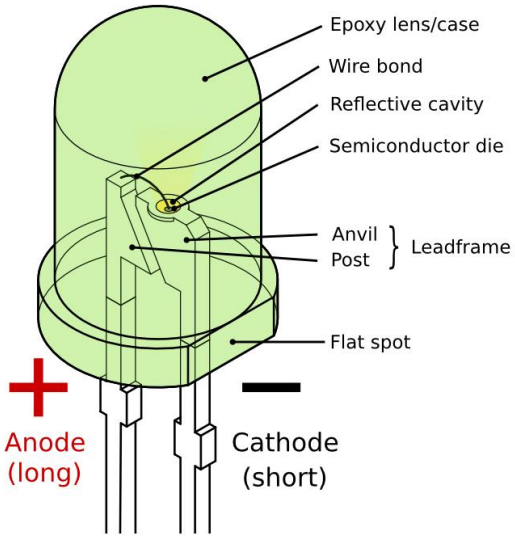
Solderless Breadboard

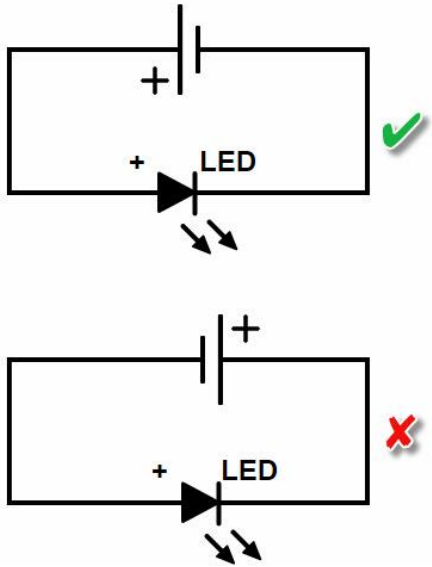


GPIO Pin Output

```
from machine import Pin



# SETUP
led = Pin(25, Pin.OUT)

# LOOP
led.toggle()
```

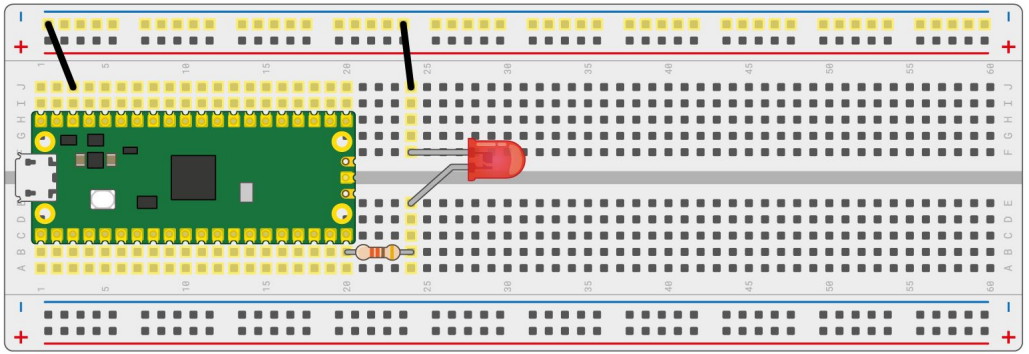
Light Emitting Diode (LED)



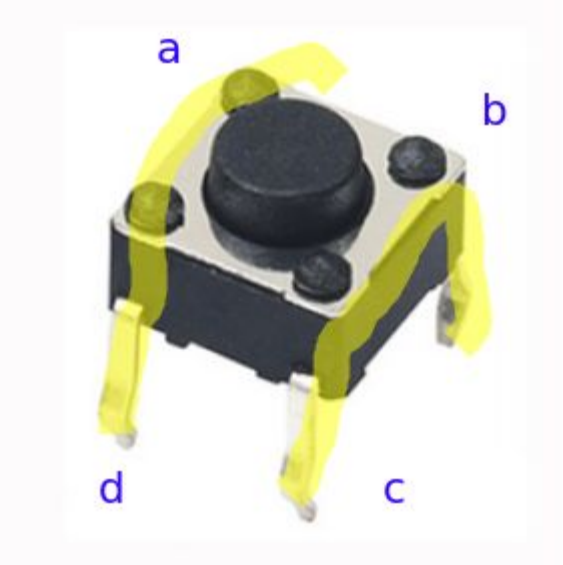


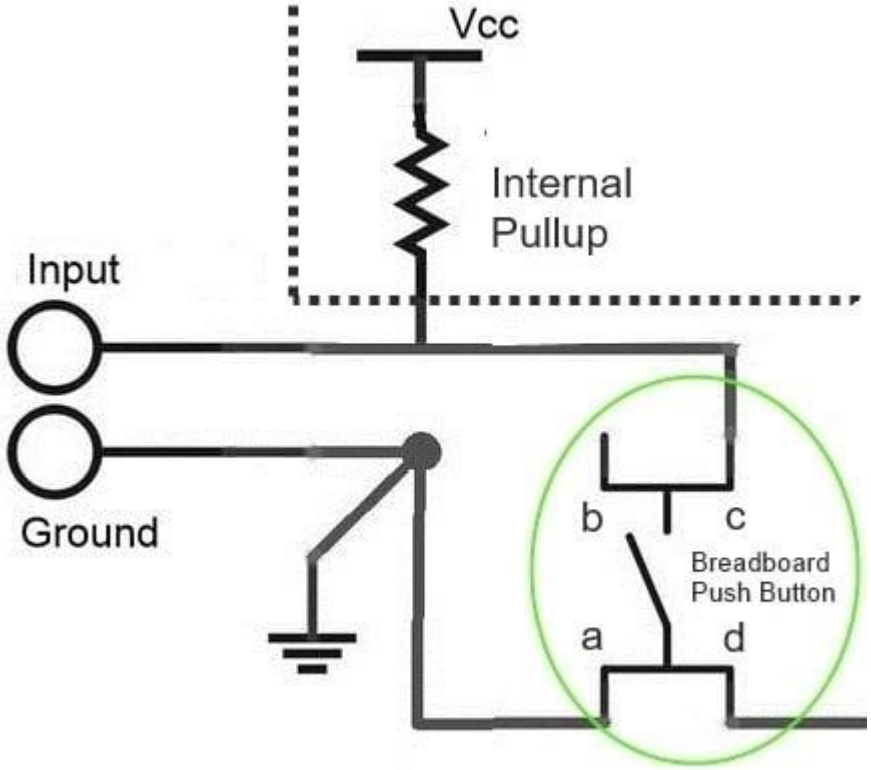
GPIO Pin Output

```
import time
import machine

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

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

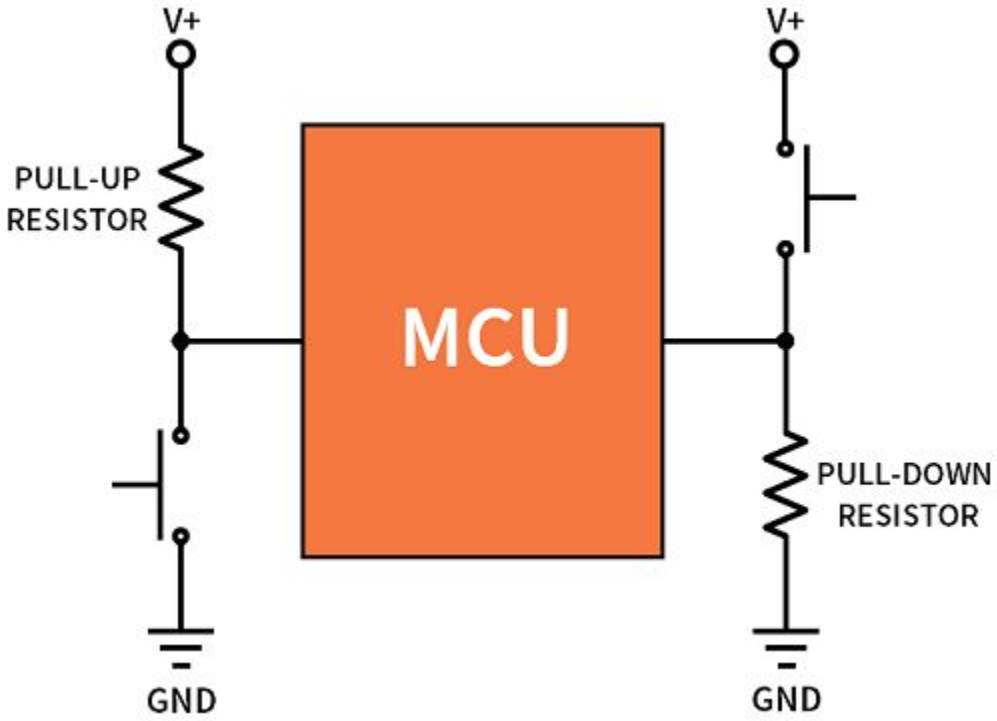


Switch Button



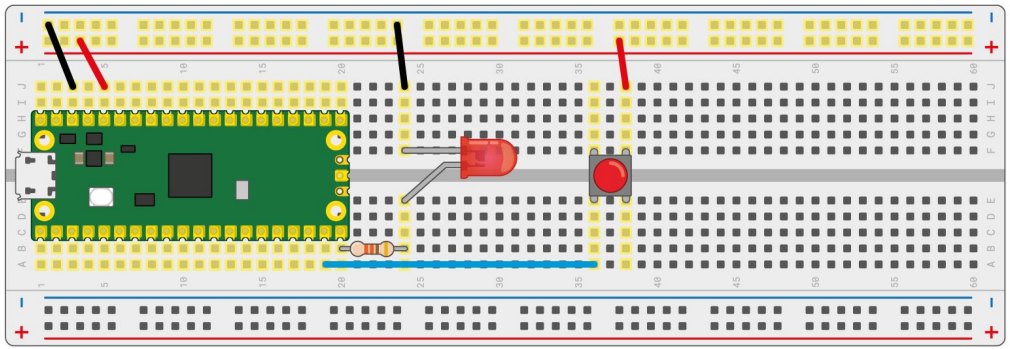


Pull-Up vs. Pull-Down Resistor

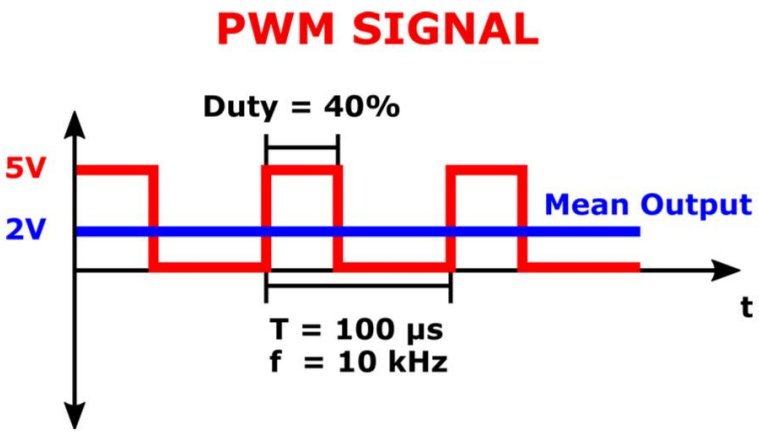


GPIO Pin Input

```
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(0.5)
```



Pulse Width Modulation (PWM)



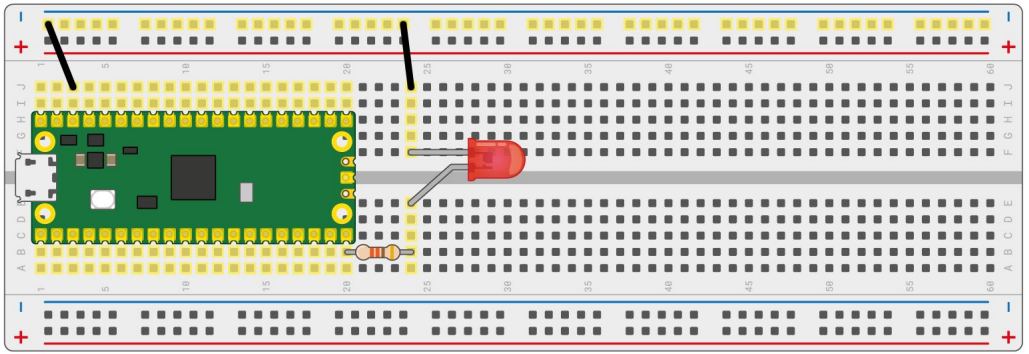
25% dimming. LED(s) are "on" 75% of the time.

50% dimming. LED(s) are "on" 50% of the time.

75% dimming. LED(s) are "on" 25% of the time.

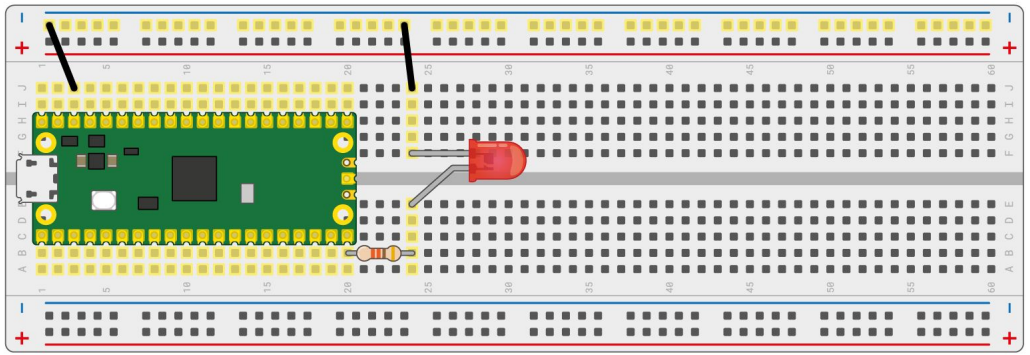
PWM Controlled LED Brightness

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

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

# LOOP
while True:
    for duty in range(65025):
        dimmer.duty_u16(duty)
        sleep(0.0001)
    for duty in range(65025, 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

```
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
```