ENGR 3421:Robotics I

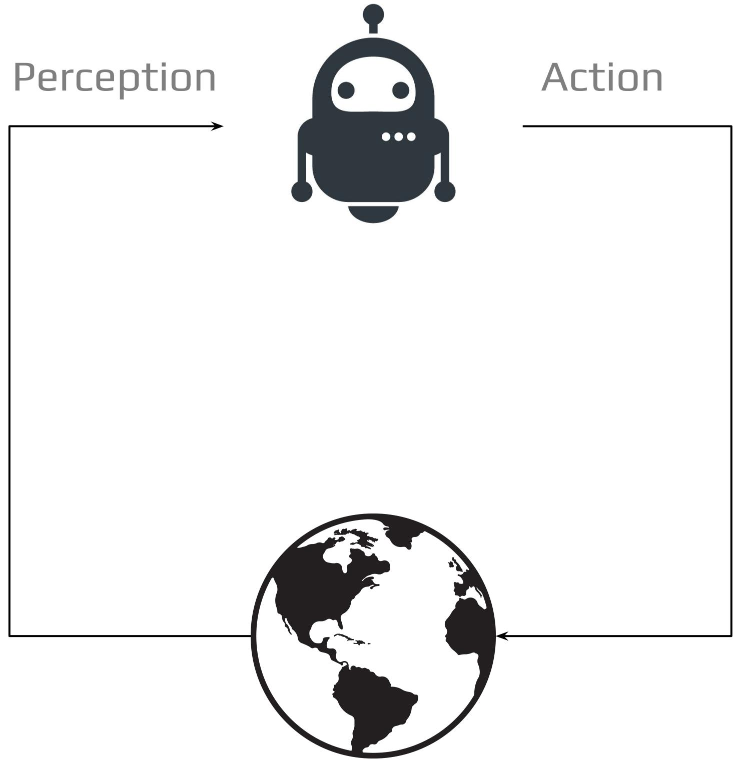
Raspberry Pi Pico

08/29/2023

Outline

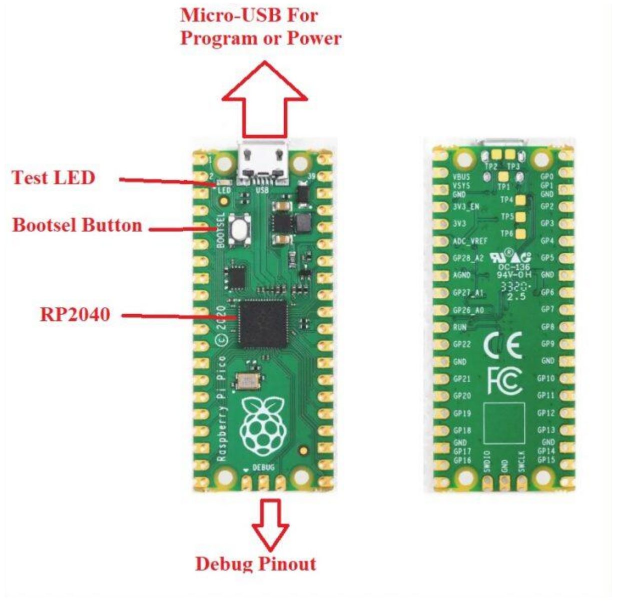
- Introduction to Raspberry Pi Pico
- MicroPython
- GPIO

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

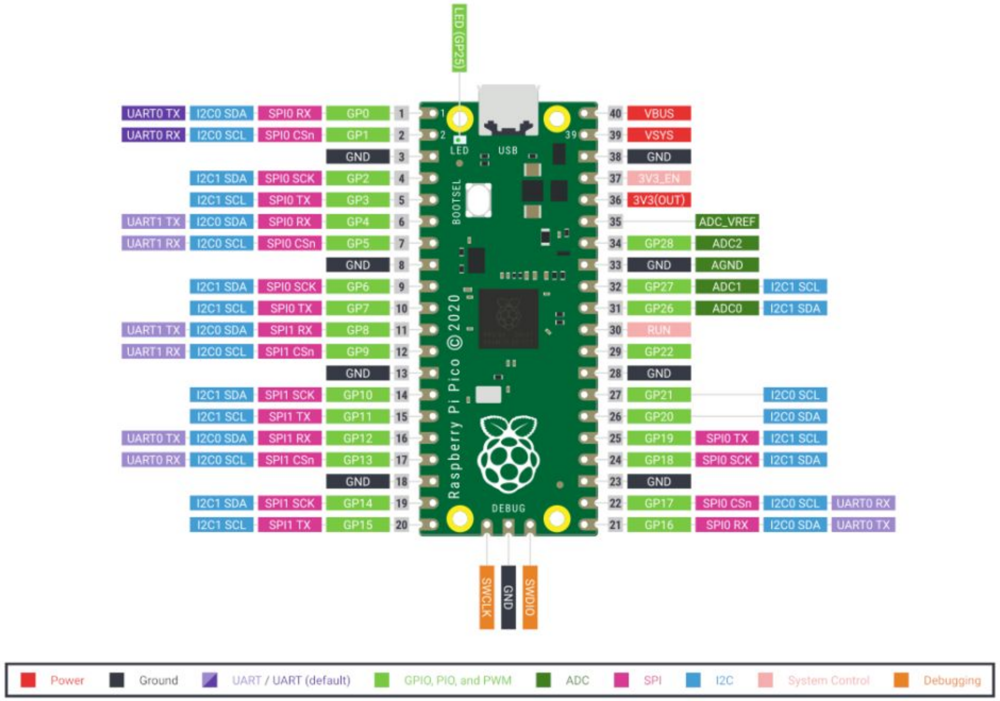
Pico Projects

- 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.

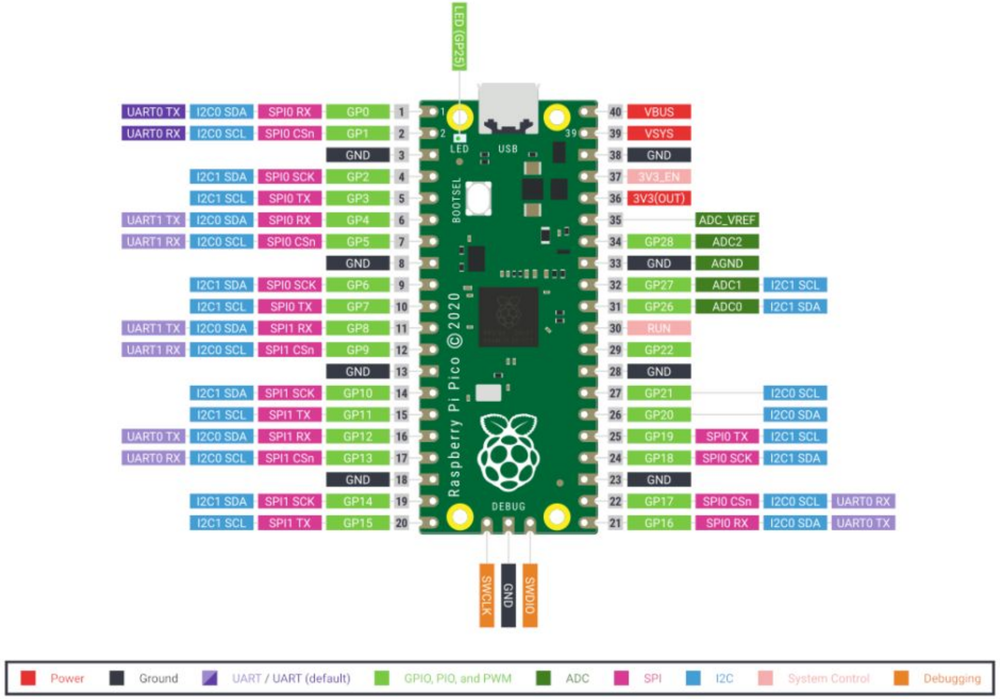


LED (GP25)
USB
LED
BOOTSEL
DEBUG
Raspberry Pi Pico ©2020
UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1
UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2
GND | 3
I2C1 SDA | SPI0 SCK | GP2 | 4
I2C1 SCL | SPI0 TX | GP3 | 5
UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6
UART1 RX | I2C0 SCL | SPI0 CSn | GP5 | 7
GND | 8
I2C1 SDA | 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

Pinout - Ground Pins

There are 9 ground pins in total.

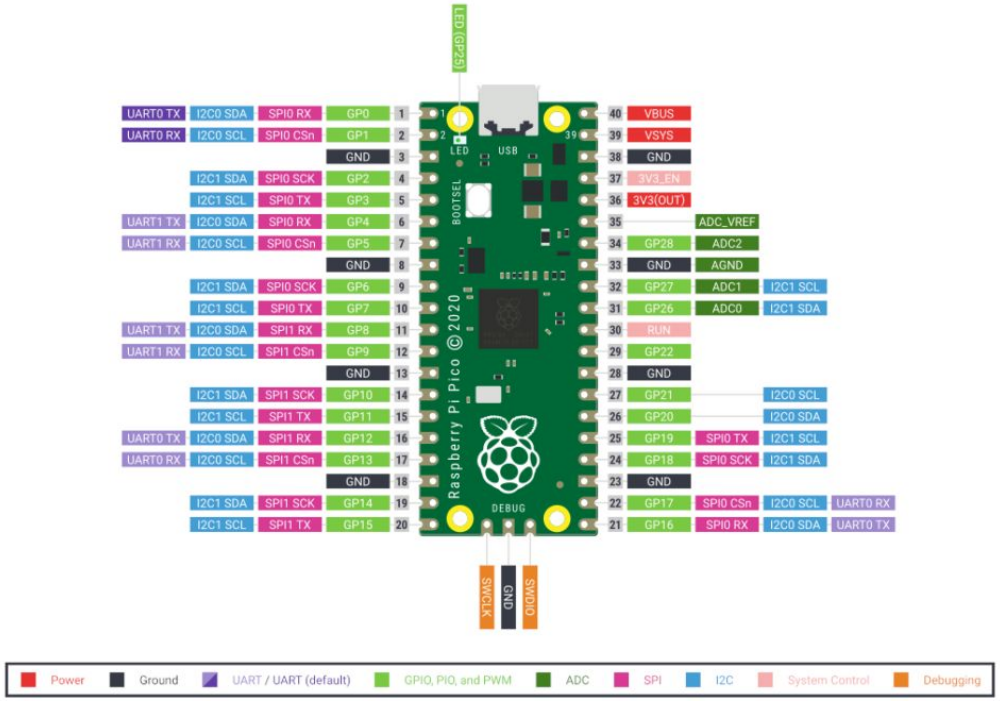
- Evenly spaced.
- Square pads.



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

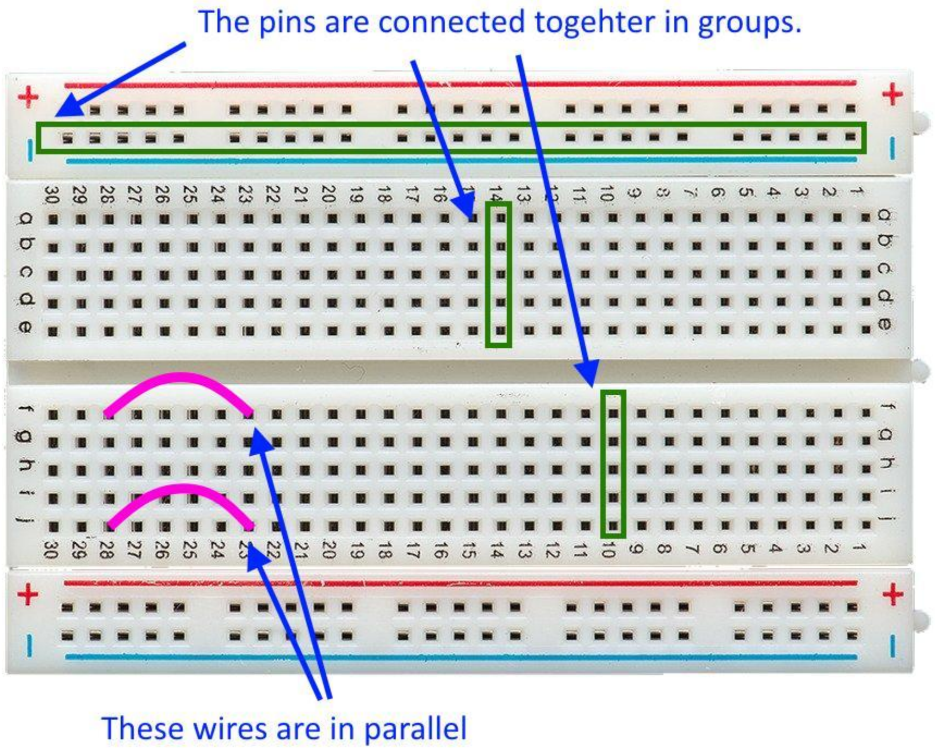


Pins Notes

- DO NOT short connect pins.
- GPIO pins use 3.3V logic for input/output. Never input 5V signals to GPIO pins.
- Max current draw is ~50mA. Don’t try to drive your motor with GPIO pins directly.
- Physical numbers vs. BCM numbers

Get Started with MicroPython

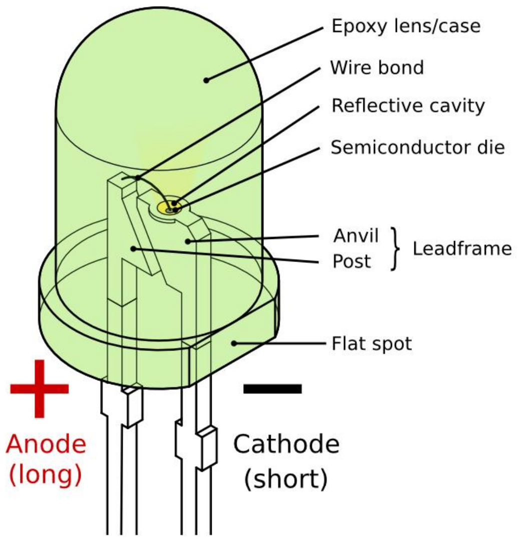
Solderless Breadboard

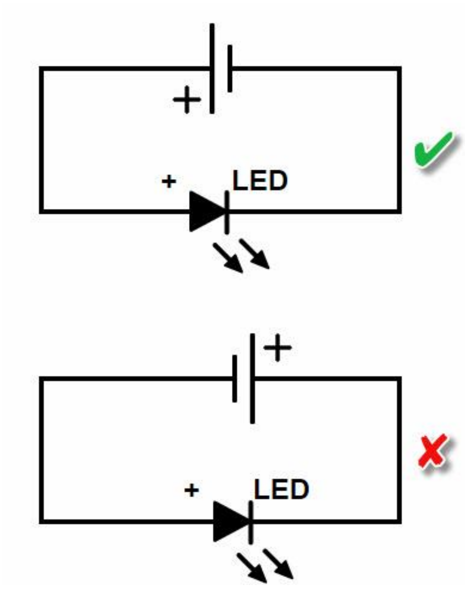


GPIO Pin Output

```
from machine import Pin



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

# LOOP
led.toggle()
```

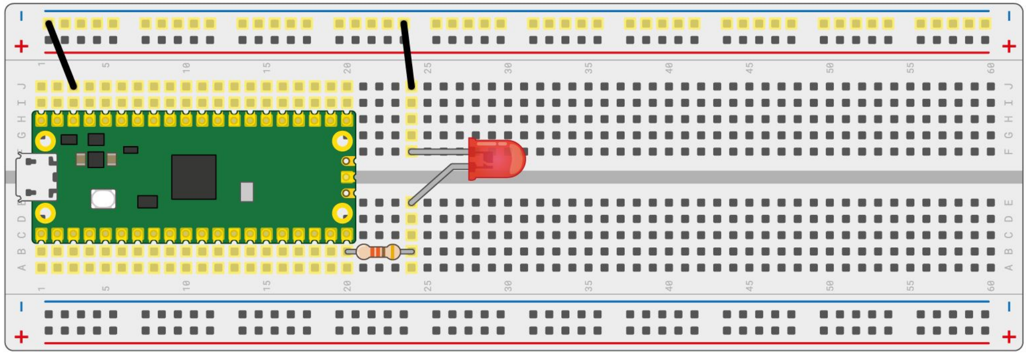
Light Emitting Diode (LED)



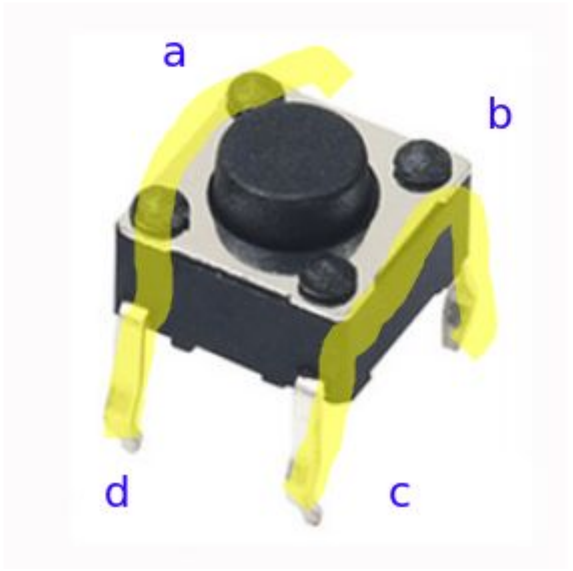


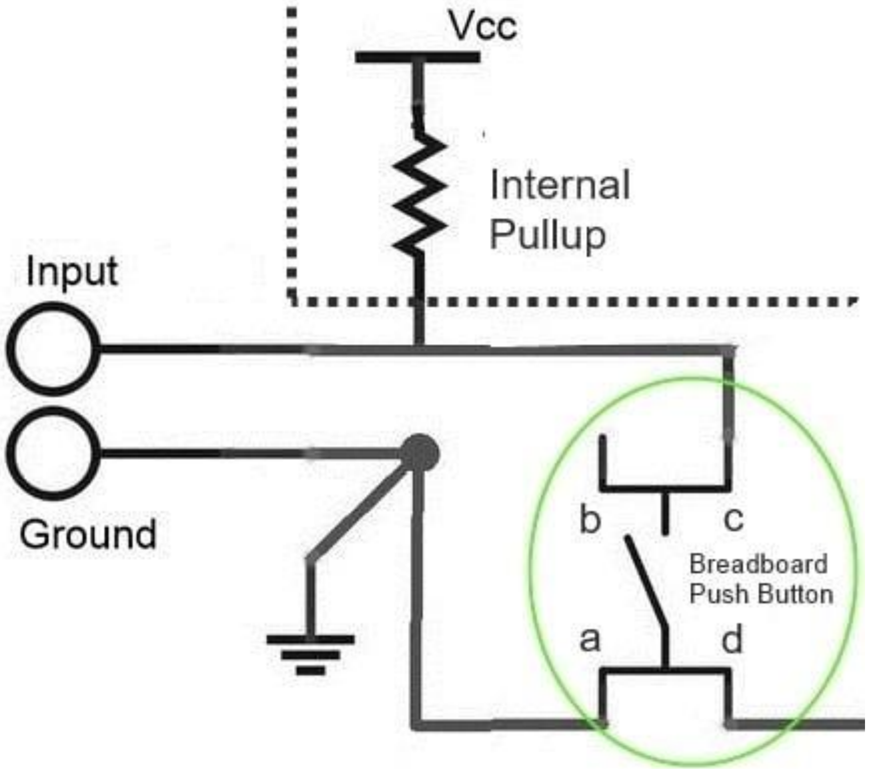
GPIO Pin Output

```
import time
import machine

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

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

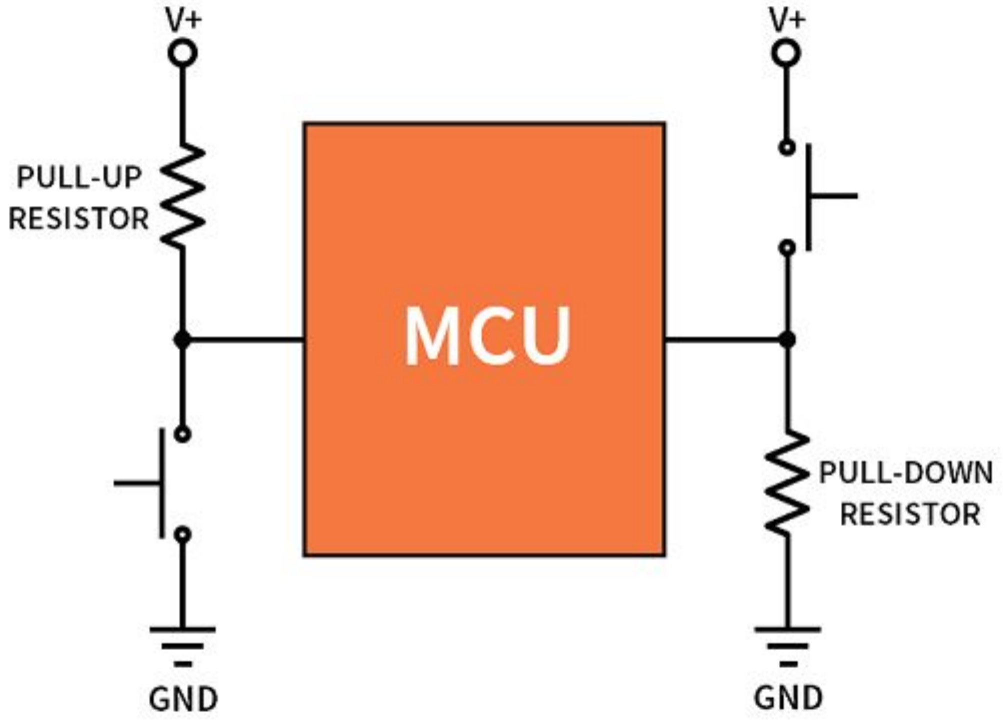


Switch Button



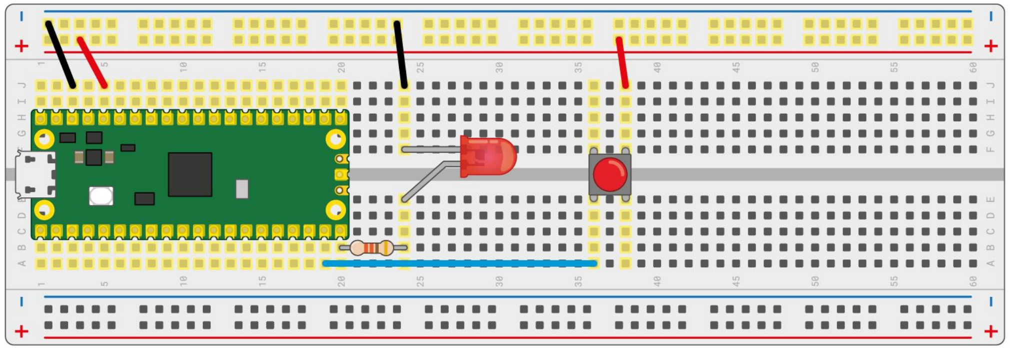


Pull-Up vs. Pull-Down Resistor

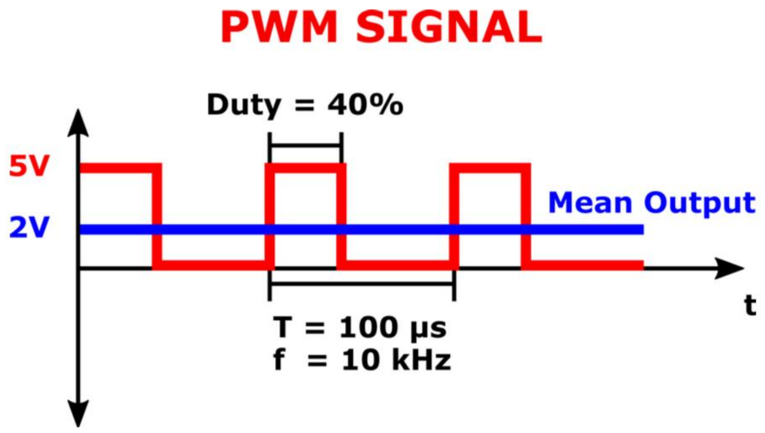


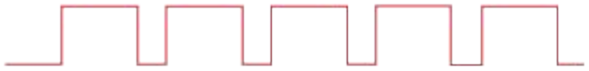
GPIO Pin Input

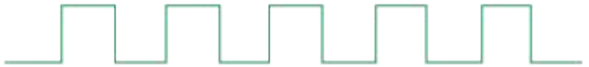
```python
from machine import Pin
from time import sleep

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

# LOOP
while True:
    if button.value() == 1:
        led.toggle()
        sleep(0.5)
```

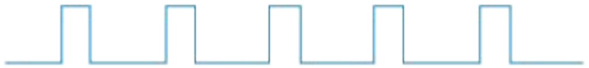


Pulse Width Modulation (PWM)



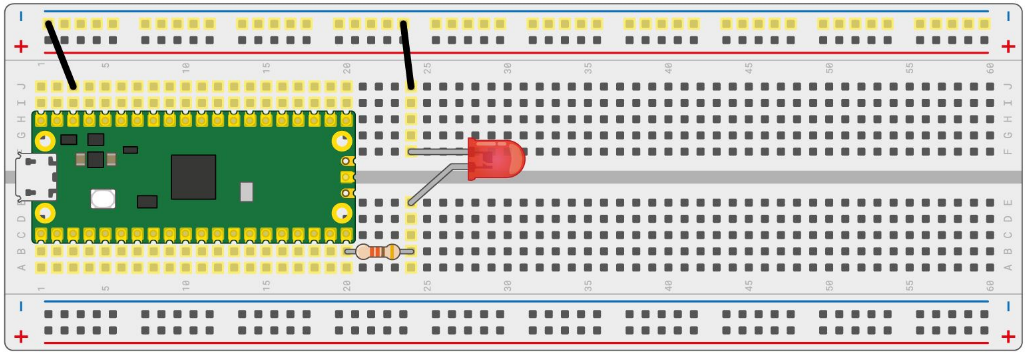

25% dimming. LED(s) are "on" 75% of the time.


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


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

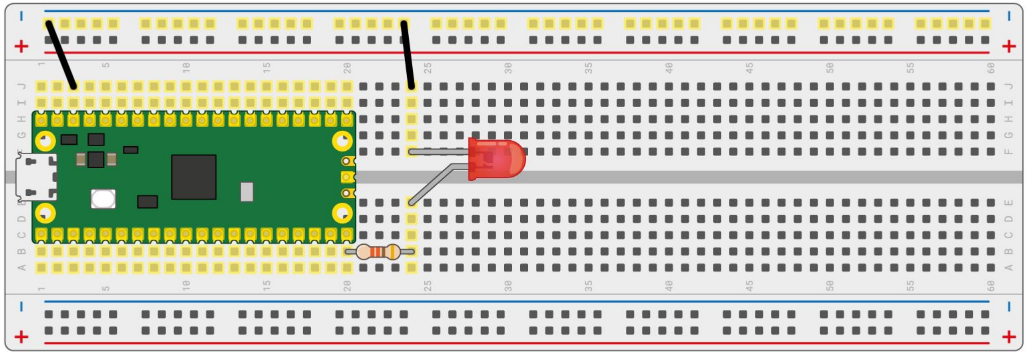
PWM Controlled LED Brightness

```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(65535):
        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
```



Interruption

```
from machine import Pin
from time import sleep

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

def toggle_led(whatever):
    led.toggle()

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

# LOOP
```