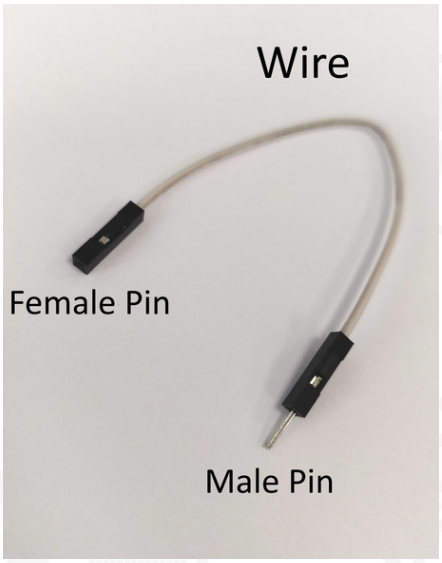
Button controlled blinking light

The basic components you will use:

Wires and Pins



Pin headers

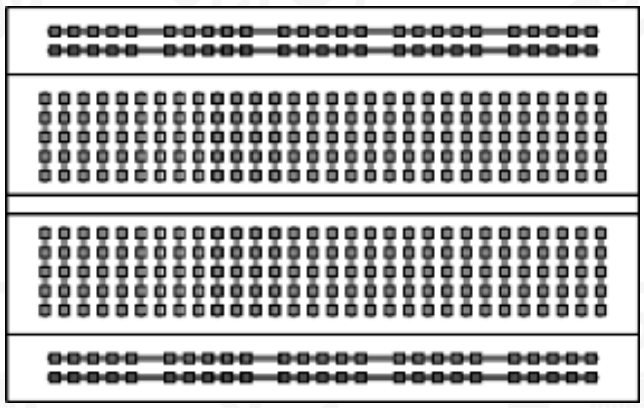
Male pin headers

Female pin headers

Breadboards

An array of female pins connected in an optimal way for circuit prototyping

Squares represent female pins, the lines represent electrical connections between the pins.



Electrical voltage

You can think of voltage as the electrical force pushing energy around a wire

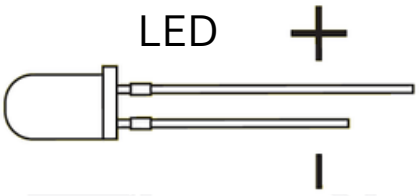
Positive voltage aka VCC, V++, 3V3, 5V+. **red** wires are associated with this.

Zero voltage aka Ground, GND. **black**, **brown** and **blue** wires are associated with this.

LED - Light Emitting diode

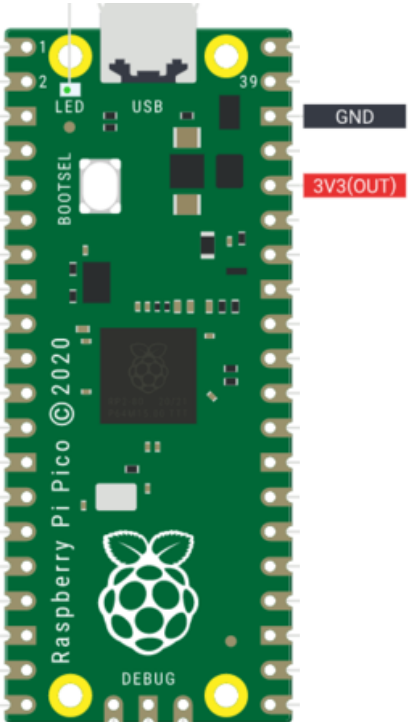
A light, it can only be powered one way, positive is the long wire, negative is the short wire

Important! - LEDs must have a resistor somewhere in its circuit otherwise it'll break



Resistors convert a percentage of electrical energy into heat which prevents the led from being overloaded.

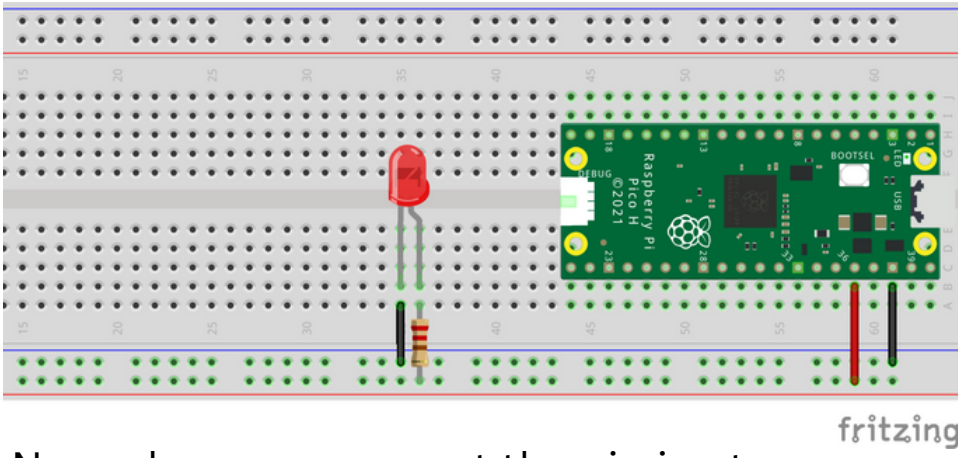
To get started lets power a LED



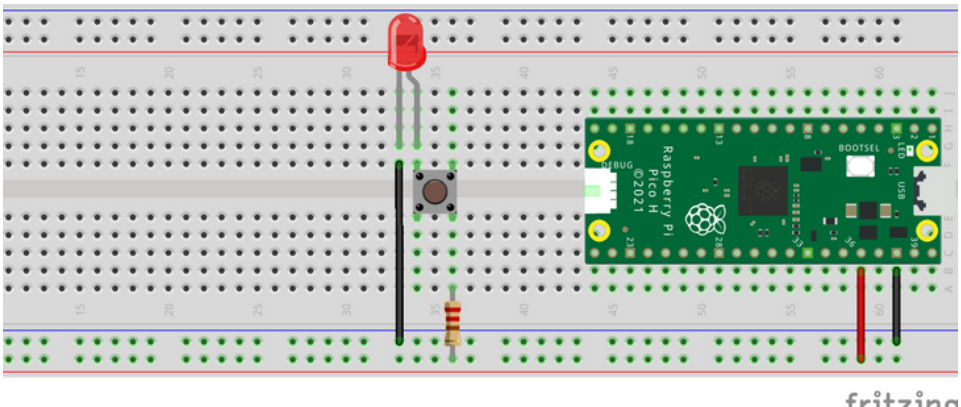
- The Pi pico on the left is a simplified diagram showing only 1 ground and 3.3 volt pin.
- Lets connect these pins to one side of the breadboard's power lines with 2 wires.

Tip: Its a good idea to use black and red colered wires respectivley for ground and power as to not get confused when you have a more complex circuit

- Wire up an LED as shown bellow, making sure the possitive (long) side of the LED is inline with the resistor and ground (short) side of the LED is inline with the ground wire.



- Now when you connect the pi pico to a computer with a usb cable the led should light up.
- Next lets add a button to the circuit like shown below, now the led will only light up when the button is pressed.



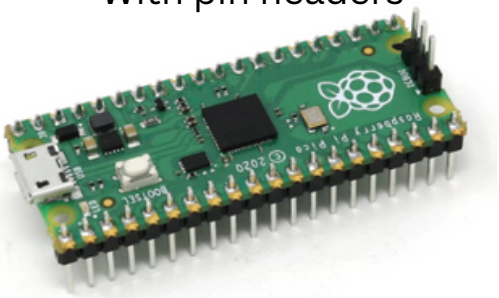
On the next page we'll do a similar thingbut using the Pi Pico to read a button press and blink the led.

What is a micro-controller?

- A Raspberry Pi Pico is an example of a micro-controller

Without pin headers

With pin headers



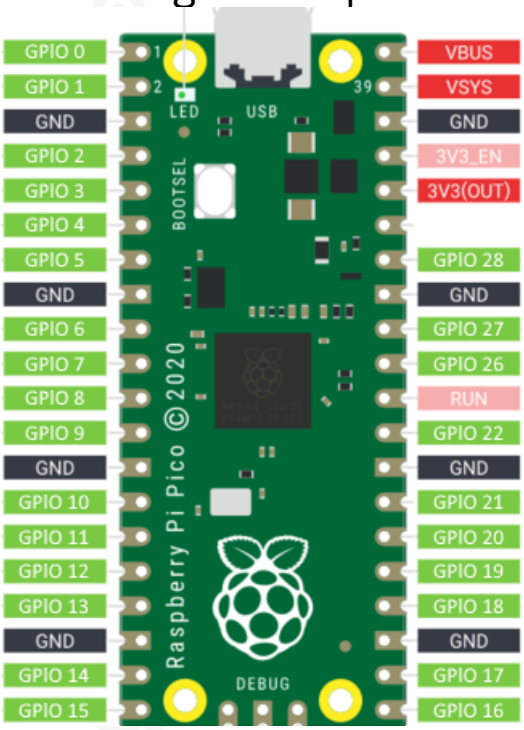
- They usually don't have an operating system and runs code directly. But because the Pi Pico is programmed in python, the code is interpreted on the Pi Pico, not compiled before hand.

- The code you write will allow you to read and write signals to pins and implement logic in-between.

- On the right you will see a full pin reference diagram (pinout diagram) for the Pi Pico

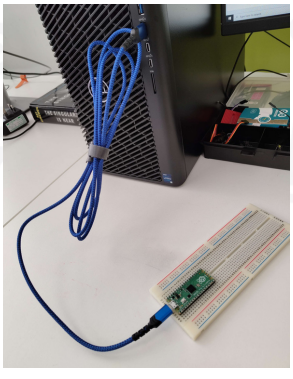
- the green labels signify **GPIO** pins, it stands for **G**eneral **P**urpose **I**nput / **O**utput pins

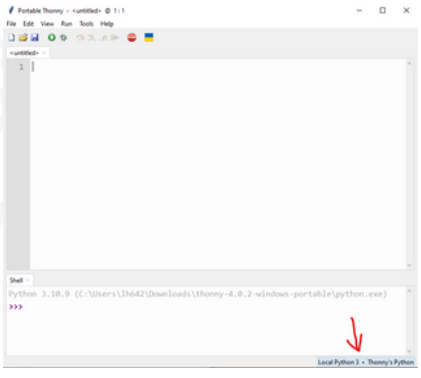
- This diagram will be your reference for interfacing with wires through code

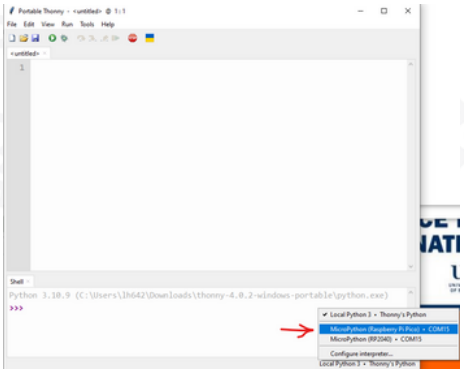


Making an LED blink

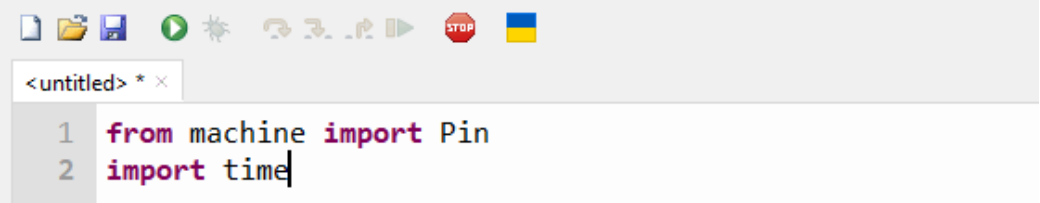
- Using the Blue USB cable, connect the Pico to a pc and open the Thonny IDE. You will need to install Portable Thonny if you want to use the FTL machines.
- With Thonny open, click the button in the bottom right corner and change the selection to "MicroPython (Raspberry Pi Pico)".

1. 

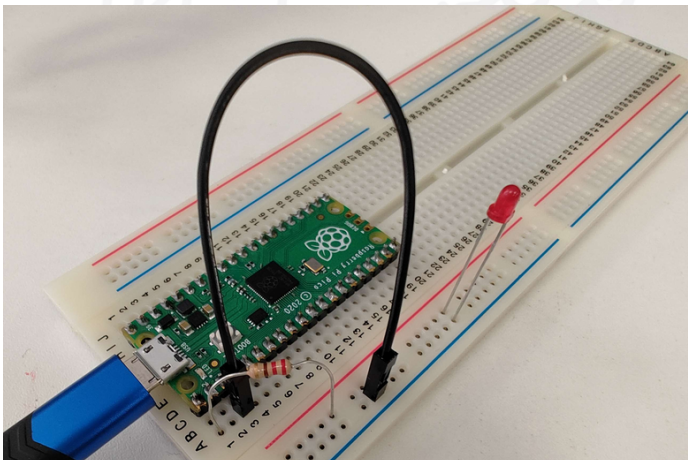
2. 

3. 

- With Thonny open, click the button in the bottom right corner and change the selection to "MicroPython (Raspberry Pi Pico)".
- Now we can start coding, begin by importing the Pin and time library with:



- Lets use GPIO pin 0 to control the LED by connecting it to the positive LED pin with a resistor. And then connect the negative side of the LED back to any GND pin on the Pico. As shown bellow:



Tip: use the pin diagram on the previous page for reference

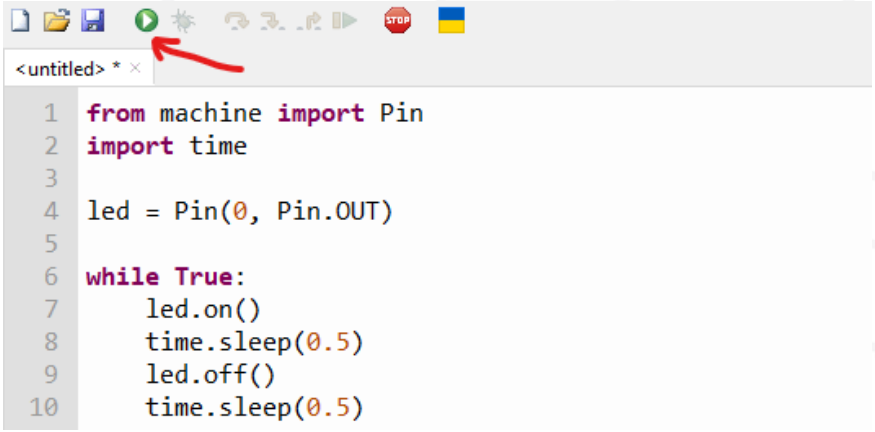
- Now for the code, we will make a variable called "led" to store a "Pin" object. Giving the object GPIO 0 and output as arguments:

```
3
4 led = Pin(0, Pin.OUT)
```

- Lets make a while loop to constantly blink the led, we can do this by making the led on, wait 0.5 seconds, led off, wait 0.5 seconds, repeat:

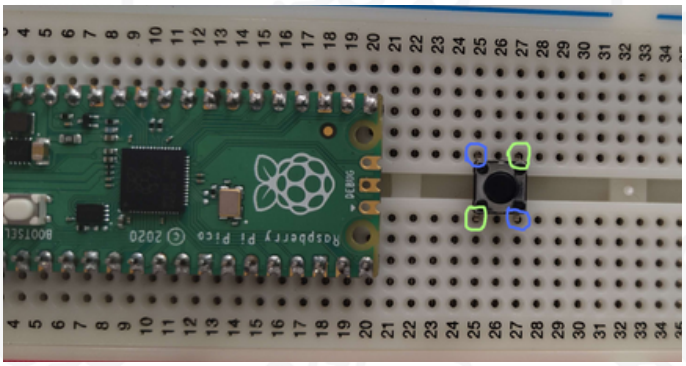
```
5
6 while True:
7     led.on()
8     time.sleep(0.5)
9     led.off()
10    time.sleep(0.5)
```

- Now run the code to see your LED blink!



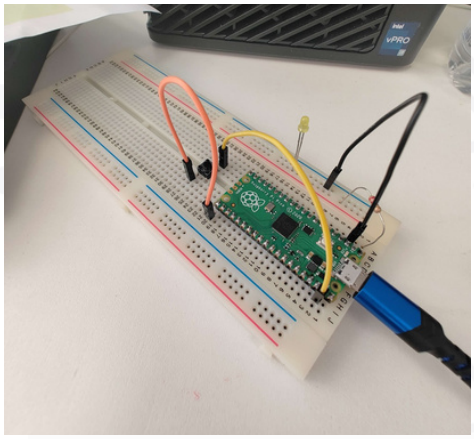
Making an Led blink on a button press:

- Now we have a blinking LED we can make it button activated!
- Place a button in the middle of the breadboard as shown below:



Diagonally opposite ends of the button are electrically connected when pressed.

- You could just put the button in series with your existing circuit but we'll do it using some code instead, first connect up a button pin to **3V3** and the other to GPIO pin 16 like shown on the left:



The additional code on line 5 and 8:

```
1  from machine import Pin
2  import time
3
4  led = Pin(0, Pin.OUT)
5  button = Pin(16, Pin.IN, Pin.PULL_DOWN)
6
7  while True:
8      if button.value() == 1:
9          led.on()
10         time.sleep(0.5)
11         led.off()
12         time.sleep(0.5)
13
```

The Pin.PULL_DOWN is required for the button because GPIO pins can sometimes have a residual voltage, so this code forces the pin to be pulled down to 0V unless connected to a proper power source.

Use the machine.Pin class:

```
from machine import Pin

p0 = Pin(0, Pin.OUT)    # create output pin on GPIO0
p0.on()                 # set pin to "on" (high) level
p0.off()                # set pin to "off" (low) level
p0.value(1)             # set pin to on/high

p2 = Pin(2, Pin.IN)     # create input pin on GPIO2
print(p2.value())       # get value, 0 or 1

p4 = Pin(4, Pin.IN, Pin.PULL_UP) # enable internal pull-up resistor
p5 = Pin(5, Pin.OUT, value=1) # set pin high on creation
```

```
import time

time.sleep(1)           # sleep for 1 second
time.sleep_ms(500)      # sleep for 500 milliseconds
time.sleep_us(10)       # sleep for 10 microseconds
start = time.ticks_ms() # get millisecond counter
delta = time.ticks_diff(time.ticks_ms(), start) # compute time differen
```

Pi Pico Code Documentation

Worksheet produced by Lukas Hastings for use by HackSussex
https://github.com/supersand21/Robotics-Workshop