Féidearthachtaí as Cuimse
Infinite Possibilities

Week 2
Raspberry Pi Pico

Fundamentals of IoT

Dr. Eoin Rogers (eoin.rogers@tudublin.ie)

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Lesson Outline

- Introducing the Pico WH (our first piece of hardware)
- Basic pinout for the Pico WH
- Programming the Pico WH using MicroPython

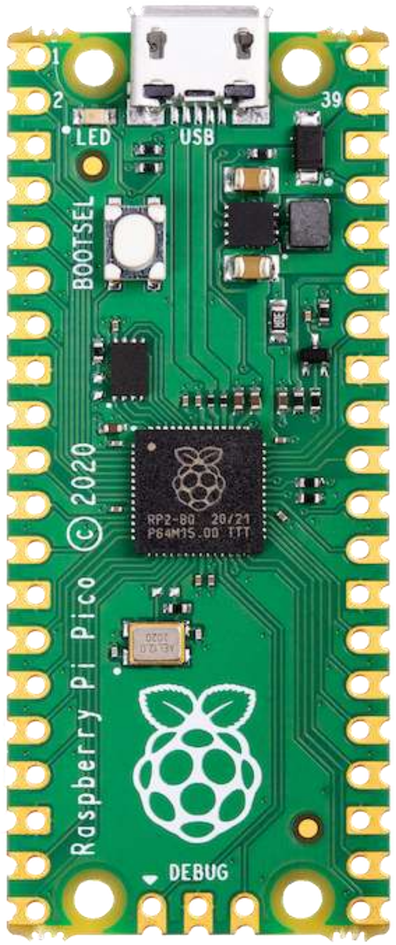
Raspberry Pi Pico WH

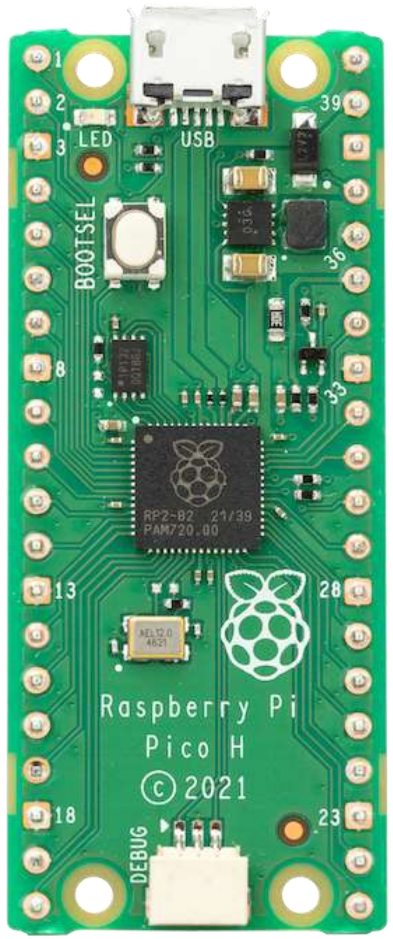
Our first piece of hardware!

Specs

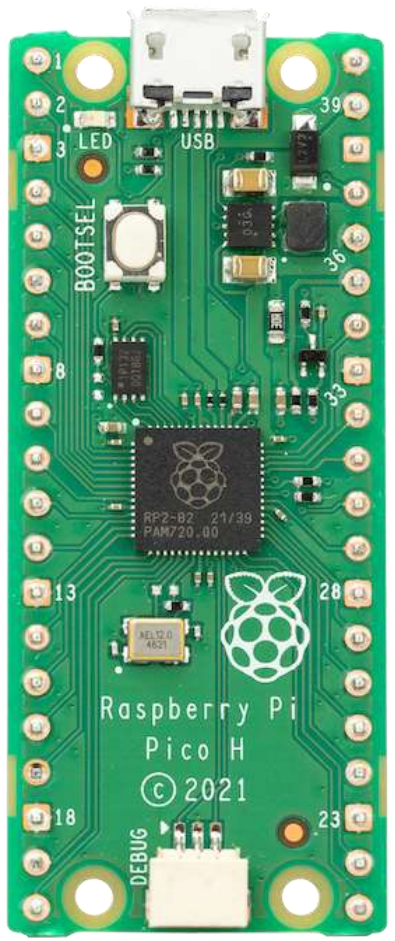
- RP2040 microcontroller (dual Cortex M0 @ 133MHz, 256KB RAM)
- 2MB of flash memory (disk space)
- 26 GPIO pins, SPI and I^{2}C support
- Software FP support(!)

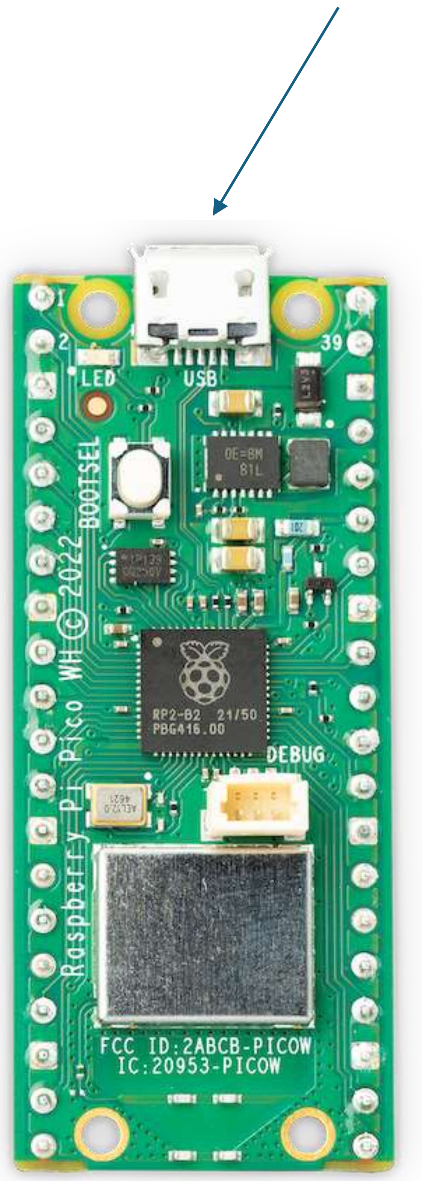
Pico 1 family





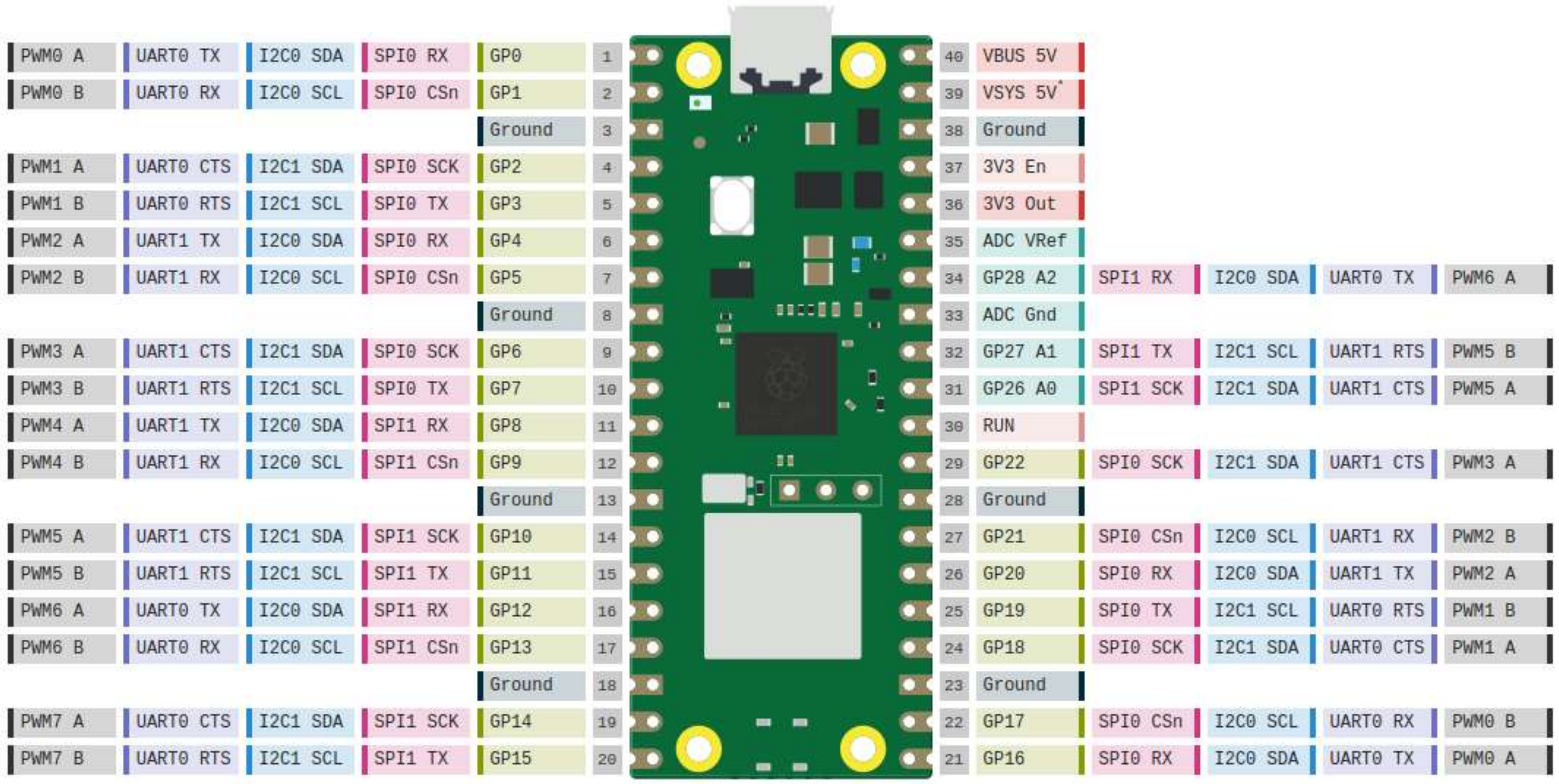
Pico 1 family





Raspberry Pi Pico WH

Pico W(H) pinout

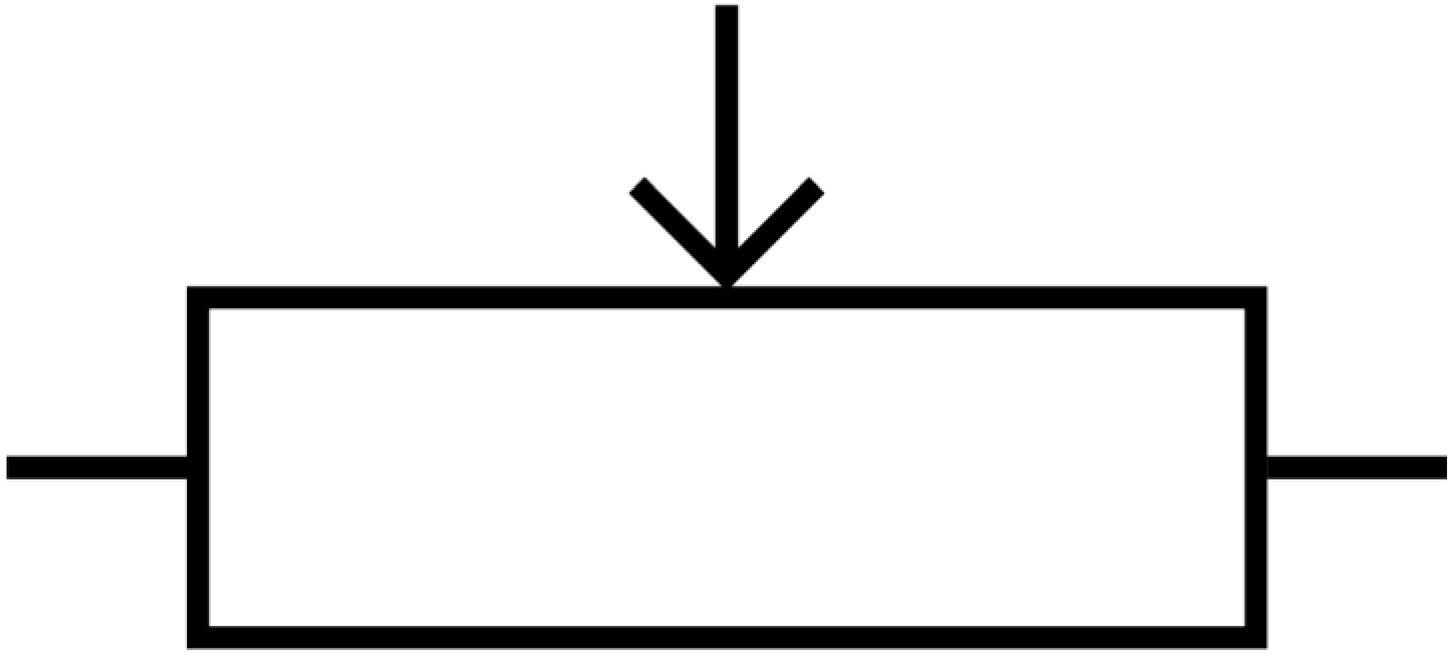


PWM0 A	UART0 TX	I2C0 SDA	SPI0 RX	GP0	1	40	VBUS 5V				
PWM0 B	UART0 RX	I2C0 SCL	SPI0 CSn	GP1	2	39	VSYS 5V*				
				Ground	3	38	Ground				
PWM1 A	UART0 CTS	I2C1 SDA	SPI0 SCK	GP2	4	37	3V3 En				
PWM1 B	UART0 RTS	I2C1 SCL	SPI0 TX	GP3	5	36	3V3 Out				
PWM2 A	UART1 TX	I2C0 SDA	SPI0 RX	GP4	6	35	ADC VRef				
PWM2 B	UART1 RX	I2C0 SCL	SPI0 CSn	GP5	7	34	GP28 A2	SPI1 RX	I2C0 SDA	UART0 TX	PWM6 A
				Ground	8	33	ADC Gnd				
PWM3 A	UART1 CTS	I2C1 SDA	SPI0 SCK	GP6	9	32	GP27 A1	SPI1 TX	I2C1 SCL	UART1 RTS	PWM5 B
PWM3 B	UART1 RTS	I2C1 SCL	SPI0 TX	GP7	10	31	GP26 A0	SPI1 SCK	I2C1 SDA	UART1 CTS	PWM5 A
PWM4 A	UART1 TX	I2C0 SDA	SPI1 RX	GP8	11	30	RUN				
PWM4 B	UART1 RX	I2C0 SCL	SPI1 CSn	GP9	12	29	GP22	SPI0 SCK	I2C1 SDA	UART1 CTS	PWM3 A
				Ground	13	28	Ground				
PWM5 A	UART1 CTS	I2C1 SDA	SPI1 SCK	GP10	14	27	GP21	SPI0 CSn	I2C0 SCL	UART1 RX	PWM2 B
PWM5 B	UART1 RTS	I2C1 SCL	SPI1 TX	GP11	15	26	GP20	SPI0 RX	I2C0 SDA	UART1 TX	PWM2 A
PWM6 A	UART0 TX	I2C0 SDA	SPI1 RX	GP12	16	25	GP19	SPI0 TX	I2C1 SCL	UART0 RTS	PWM1 B
PWM6 B	UART0 RX	I2C0 SCL	SPI1 CSn	GP13	17	24	GP18	SPI0 SCK	I2C1 SDA	UART0 CTS	PWM1 A
				Ground	18	23	Ground				
PWM7 A	UART0 CTS	I2C1 SDA	SPI1 SCK	GP14	19	22	GP17	SPI0 CSn	I2C0 SCL	UART0 RX	PWM0 B
PWM7 B	UART0 RTS	I2C1 SCL	SPI1 TX	GP15	20	21	GP16	SPI0 RX	I2C0 SDA	UART0 TX	PWM0 A

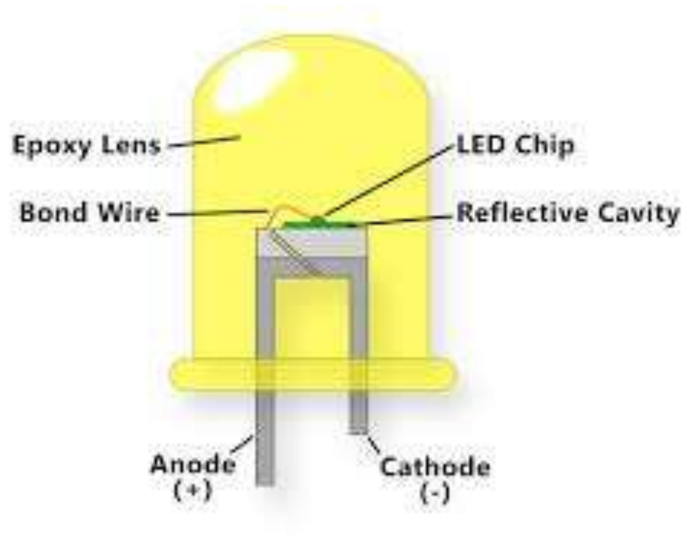
More electronic components

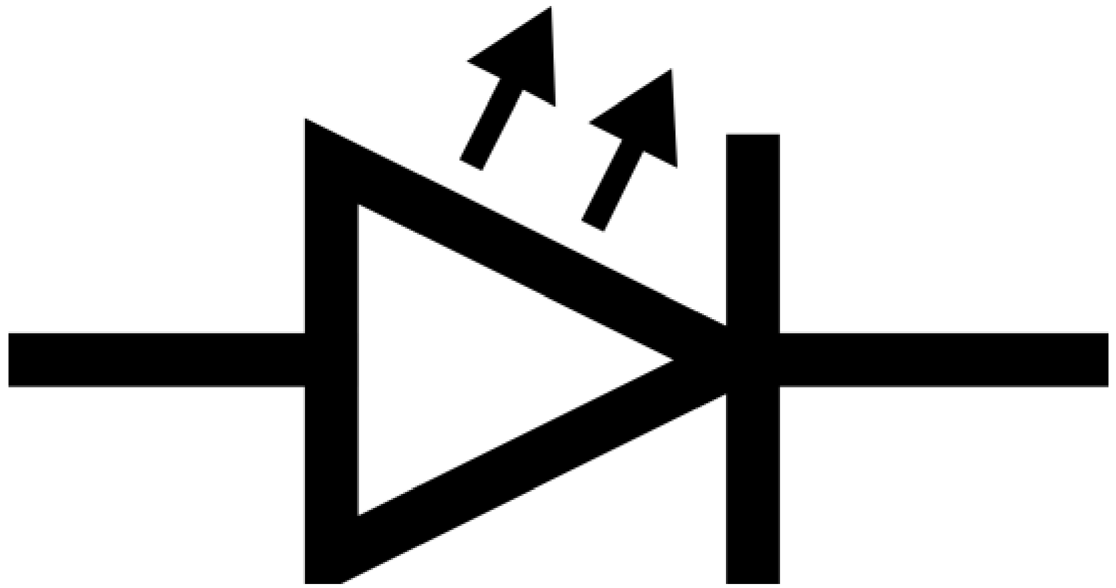
Some assorted components we can connect to the Pico

Potentiometer

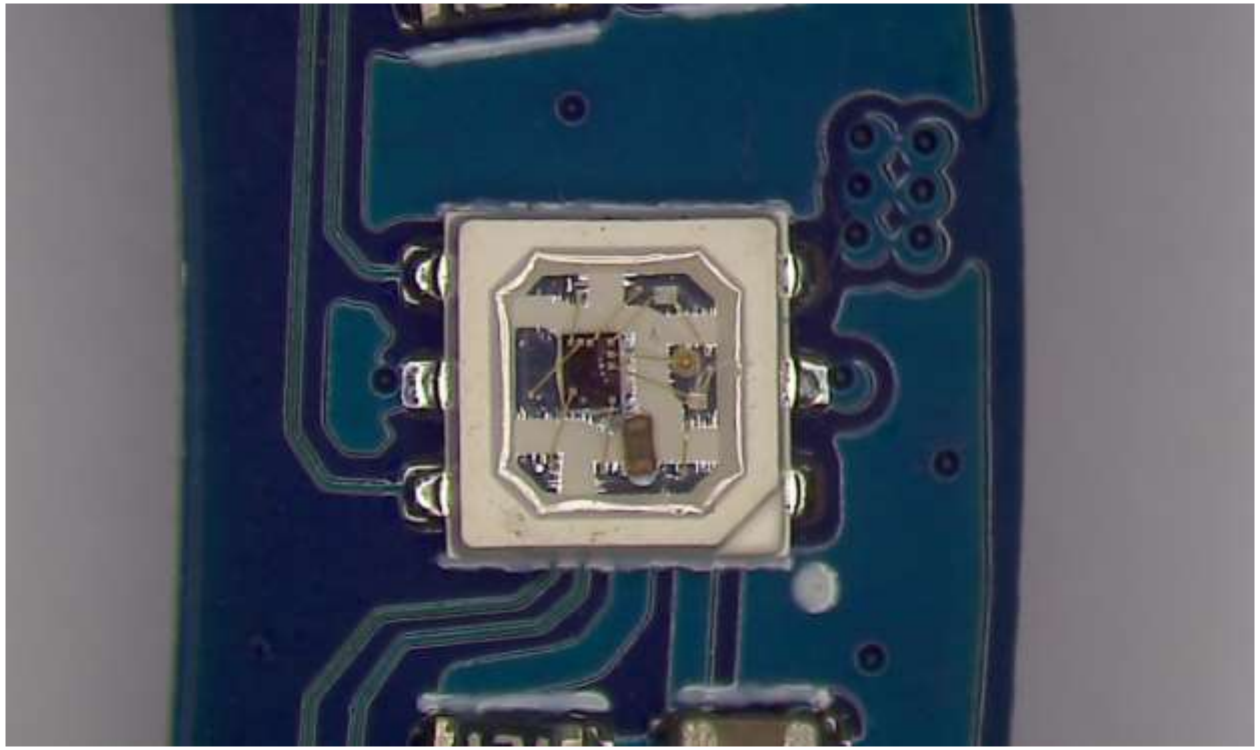


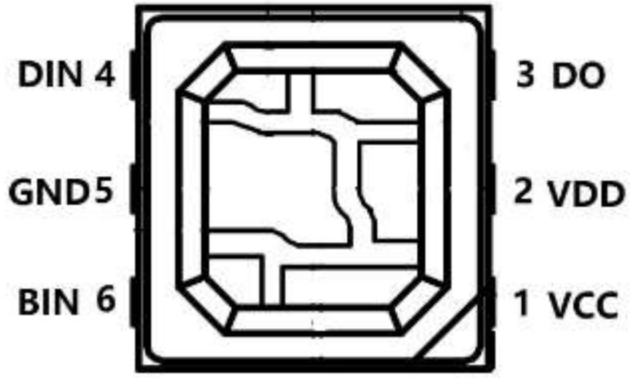
Light emitting diode (LED)



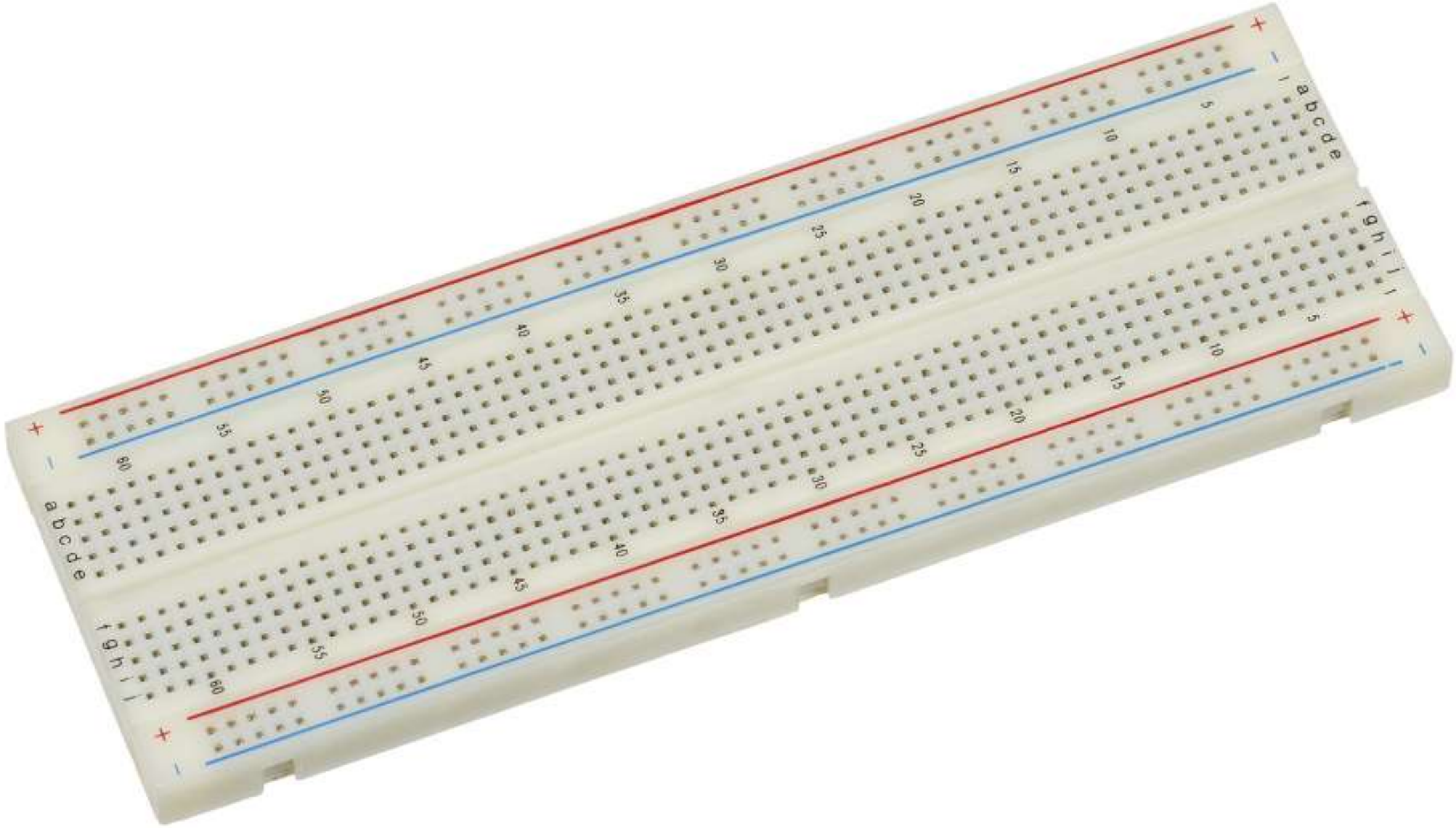


RGB LED (WS2813 mini)





Breadboard



Servomotor

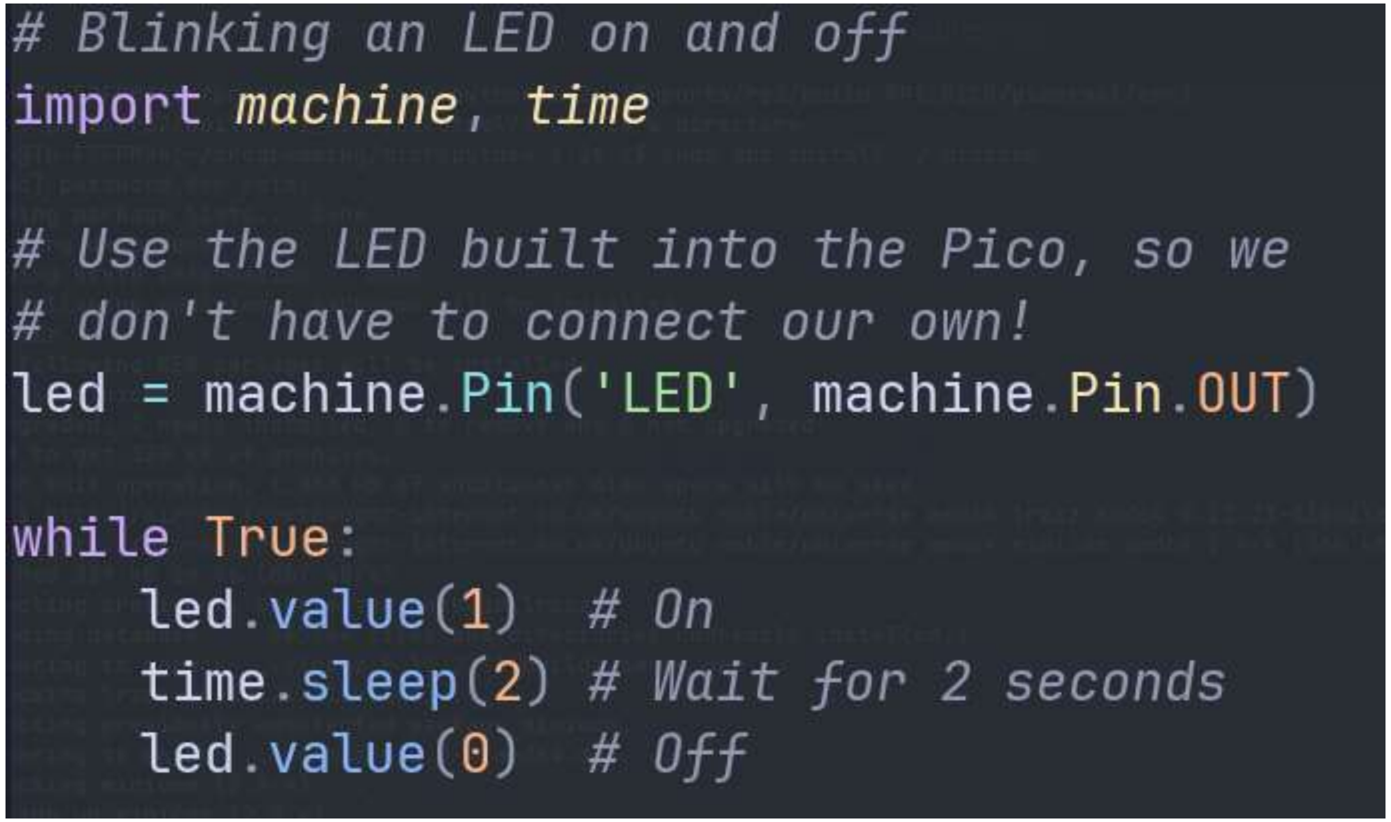
MicroPython

A dialect of Python designed for microcontrollers (µCs)

MicroPython

- Implements a large subset of Python 3.4 (and parts of 3.5)
- Runs on the metal (i.e. on systems without an operating system)
- Supports a wide range of hardware/μCs(not just the Pico)
- Open source and very easy to install!

Blinking an LED on and off

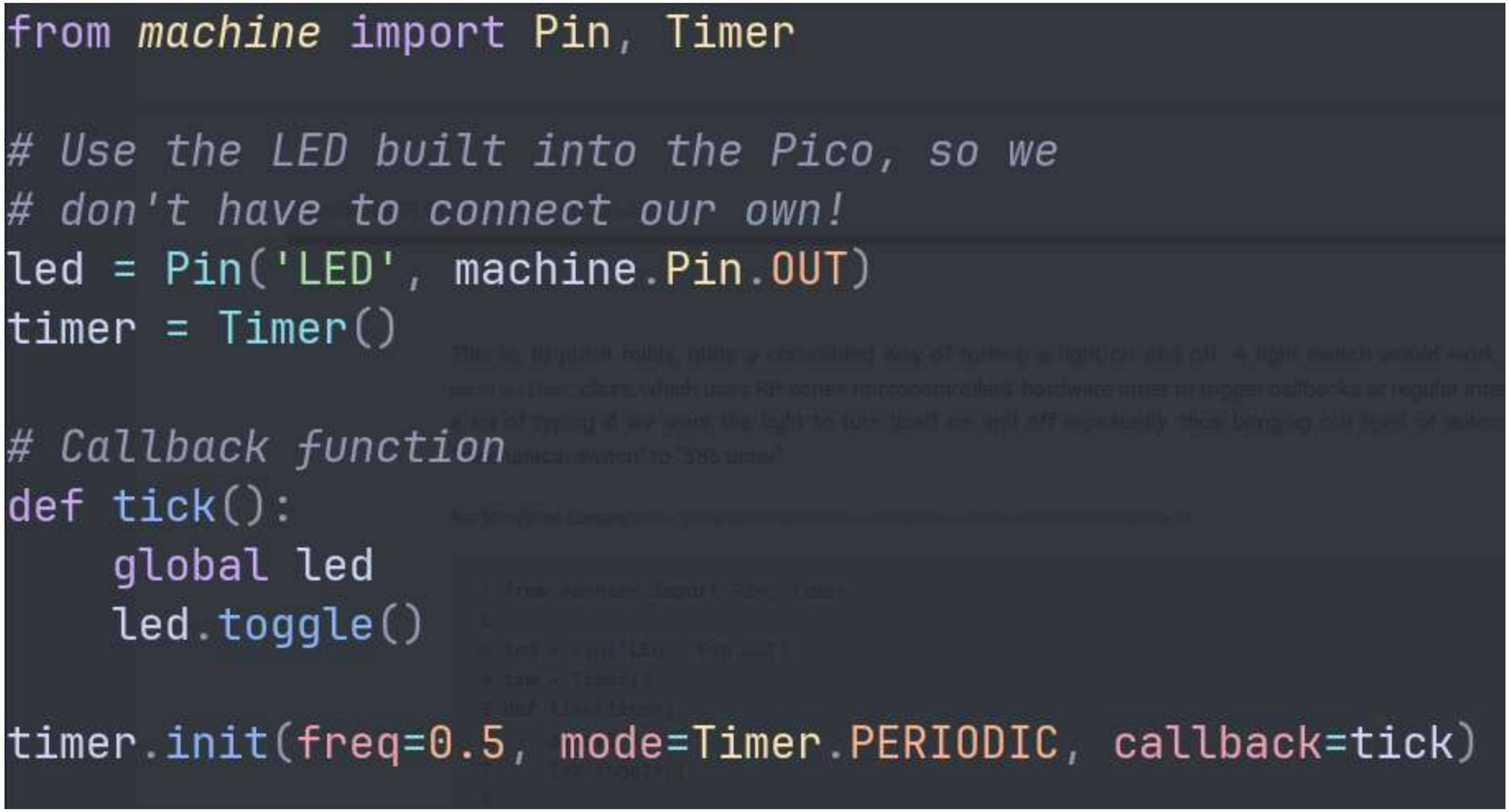


```python
# Blinking an LED on and off
import machine, time

# Use the LED built into the Pico, so we
# don't have to connect our own!
led = machine.Pin('LED', machine.Pin.OUT)

while True:
    led.value(1)  # On
    time.sleep(2) # Wait for 2 seconds
    led.value(0)  # Off
```

Better blinking script



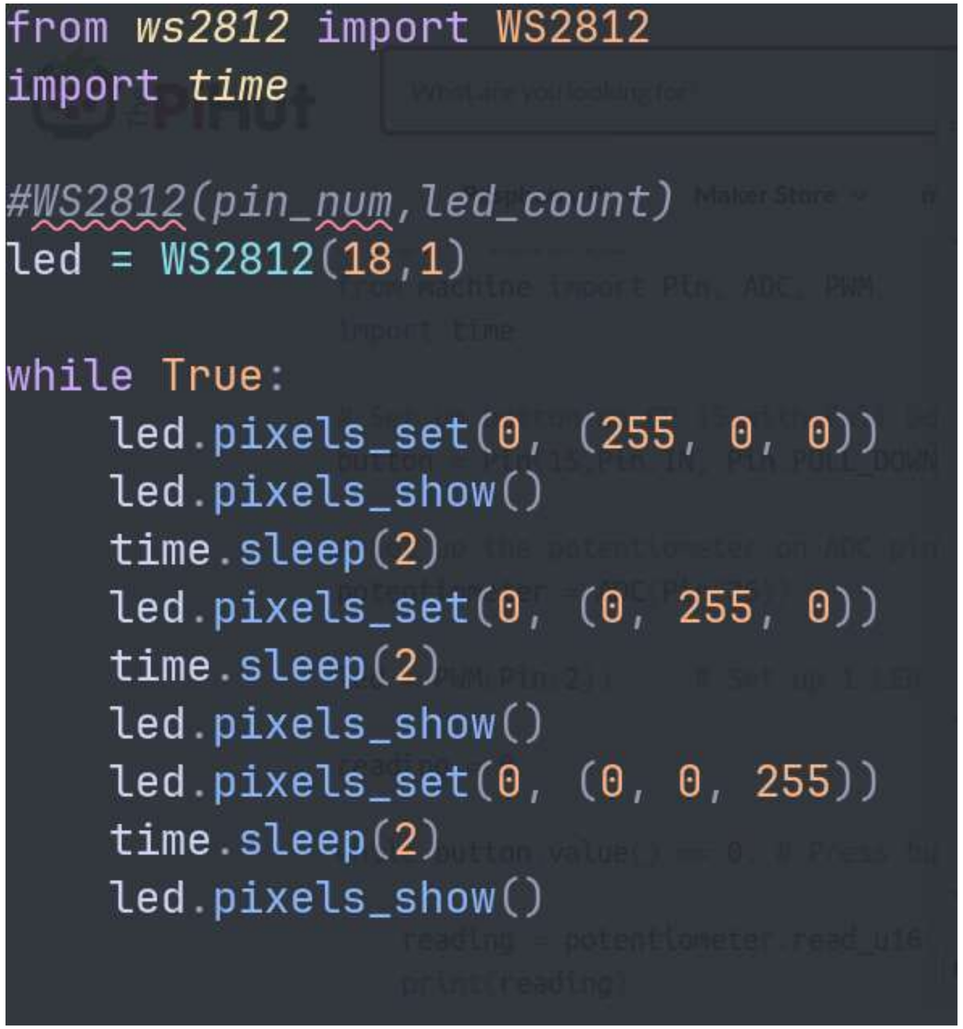
```python
from machine import Pin, Timer

# Use the LED built into the Pico, so we
# don't have to connect our own!
led = Pin('LED', machine.Pin.OUT)
timer = Timer()

# Callback function
def tick():
    global led
    led.toggle()

timer.init(freq=0.5, mode=Timer.PERIODIC, callback=tick)
```

RGB LED (WS2813)

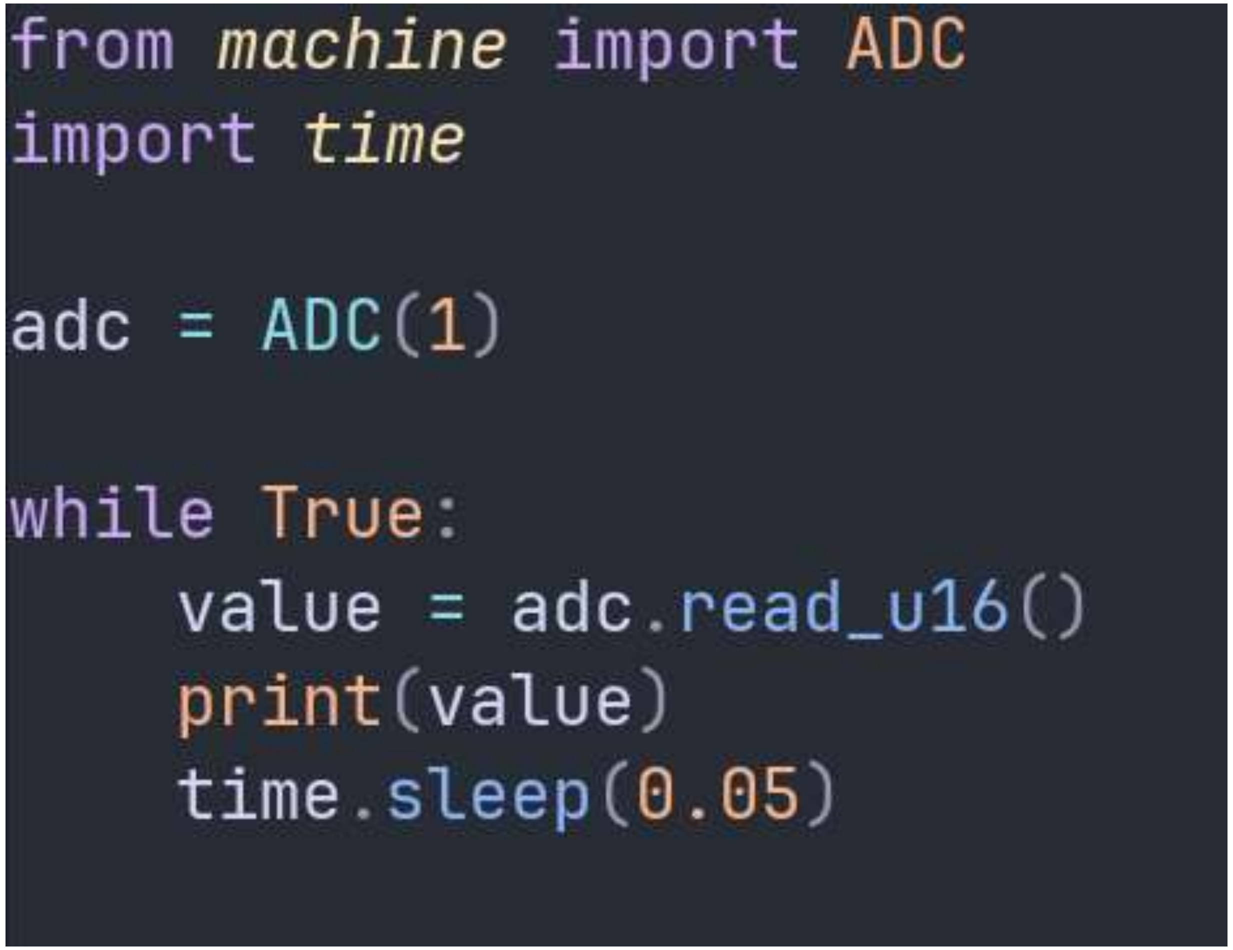


```
from ws2812 import WS2812
import time

#WS2812(pin_num,led_count)
led = WS2812(18,1)

while True:
    led.pixels_set(0, (255, 0, 0))
    led.pixels_show()
    time.sleep(2)
    led.pixels_set(0, (0, 255, 0))
    time.sleep(2)
    led.pixels_show()
    led.pixels_set(0, (0, 0, 255))
    time.sleep(2)
    led.pixels_show()
```

Reading from a potentiometer

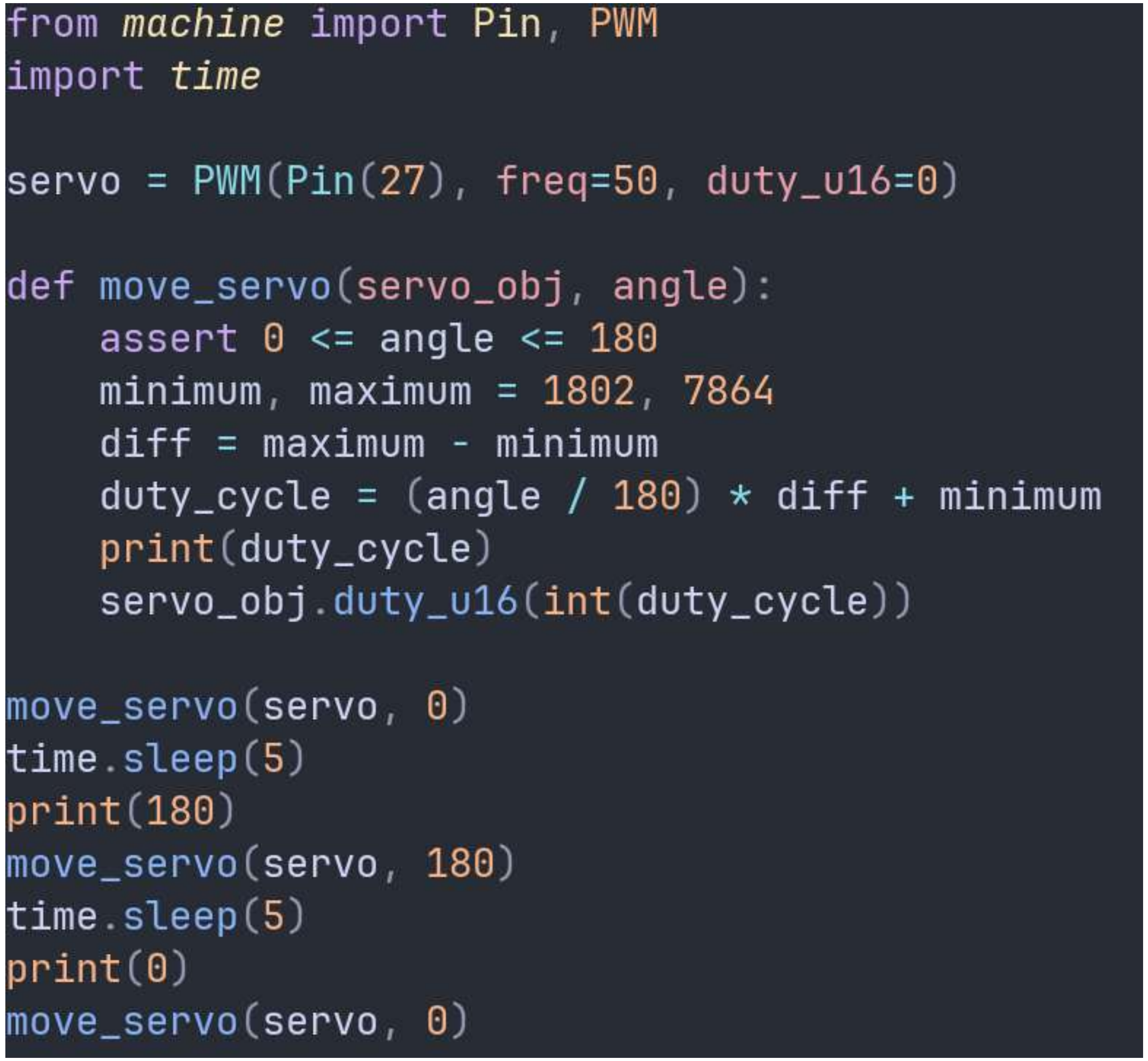


```
from machine import ADC
import time

adc = ADC(1)

while True:
    value = adc.read_u16()
    print(value)
    time.sleep(0.05)
```

Moving a servo



```
from machine import Pin, PWM
import time

servo = PWM(Pin(27), freq=50, duty_u16=0)

def move_servo(servo_obj, angle):
    assert 0 <= angle <= 180
    minimum, maximum = 1802, 7864
    diff = maximum - minimum
    duty_cycle = (angle / 180) * diff + minimum
    print(duty_cycle)
    servo_obj.duty_u16(int(duty_cycle))

move_servo(servo, 0)
time.sleep(5)
print(180)
move_servo(servo, 180)
time.sleep(5)
print(0)
move_servo(servo, 0)
```

That's all for this week

Thanks for your attention!