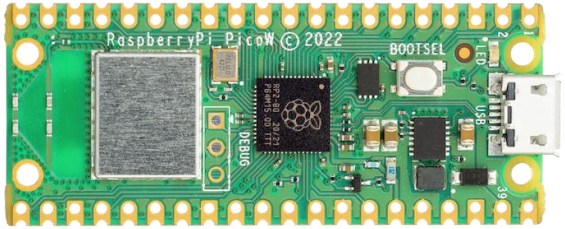
UNIVERSITI MALAYSIA PERLIS

FACULTY OF ELECTRICAL ENGINEERING & TECHNOLOGY

python

circuit python



Exploring CircuitPython on Raspberry Pi Pico-W with Thonny IDE

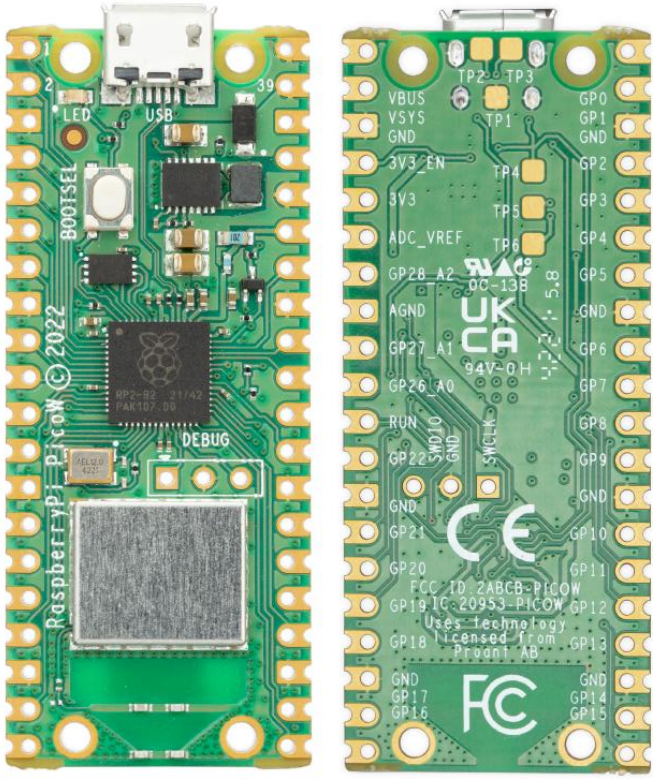
AZRALMUKMIN AZMI
azralmukmin@unimap.edu.my
019-3867034

What you will learn

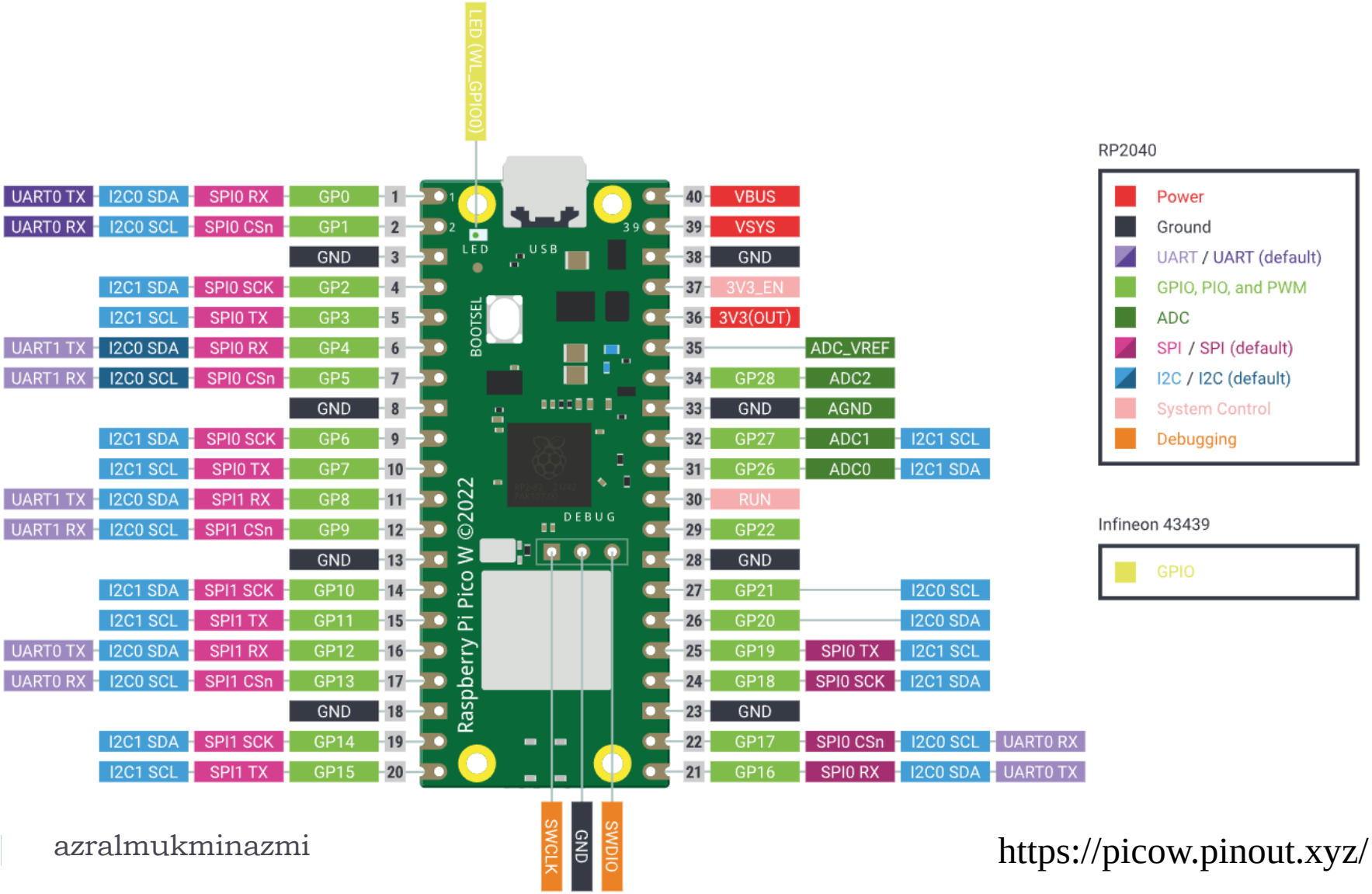
- Raspberry Pi Pico-W
- CircuitPython
- Installing CircuitPython
- Thonny IDE
- Thonny IDE – Setup CircuitPython
- Blink build-in LED
- Digital Output
- Digital Input
- Analog Input
- PWM – fixed frequency
- PWM – variable frequency
- CircuitPython Libraries
- DHT22

Raspberry Pi Pico-W

- RP2040 microcontroller chip
- Dual-core Arm Cortex M0+ processor up to 133 MHz
- Memory 264kB SRAM
- Storage 2MBytes
- 26 multi-function GPIO pins
- 2 × SPI
- 2 × I2C
- 2 × UART
- 4 × 12-bit ADC
- 16 × controllable PWM channels
- Wireless (802.11n), single-band (2.4 GHz)
- Bluetooth 5.2



Raspberry Pi Pico-W (cont)



https://picow.pinout.xyz/

CircuitPython

- Python based language designed to simplify experimenting and learning to code on low-cost microcontroller boards
- No upfront desktop downloads needed
- Simple – set board, open text editor and start code

Quick and Easy

Create a file, edit your code, save the file, and it runs immediately. There is no compiling or uploading needed.

Beginner Friendly

CircuitPython is designed with education in mind. It's an easy way to start learning how to code and you get immediate feedback from the board.

Easy Code Updates

Since your code lives on the disk drive, you can edit it whenever you like. You can even keep multiple files around for easy experimentation.

Serial Console + REPL

These features allow for live feedback from your code and interactive programming.

File Storage

The internal storage for CircuitPython makes it great for data-logging, playing audio clips, and otherwise interacting with files.

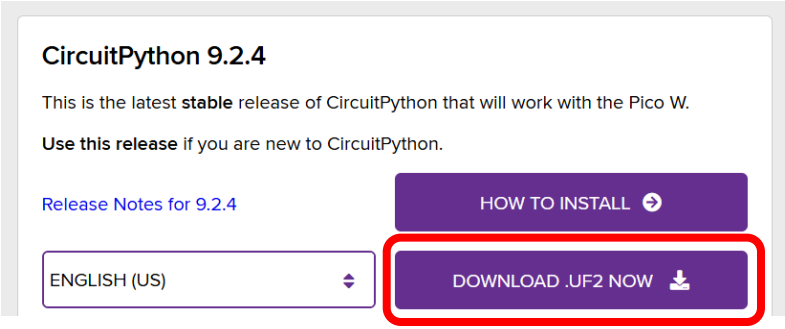
Strong Hardware Support

There are many libraries and drivers for sensors, breakout boards and other external components.

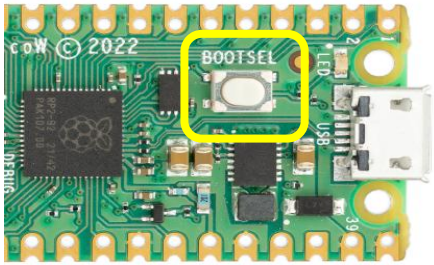
https://learn.adafruit.com/welcome-to-circuitpython

Installing CircuitPython

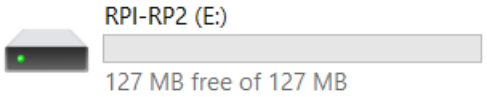
- Download **.uf2** bootloader files latest version CircuitPython 9.2.4 at https://circuitpython.org/board/raspberry_pi_pico_w/



- Hold down **BOOTSEL button** and plug Pico W into USB



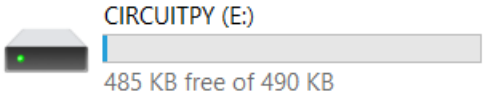
- New drive appear called **RPI-RP2**



Installing CircuitPython (cont)

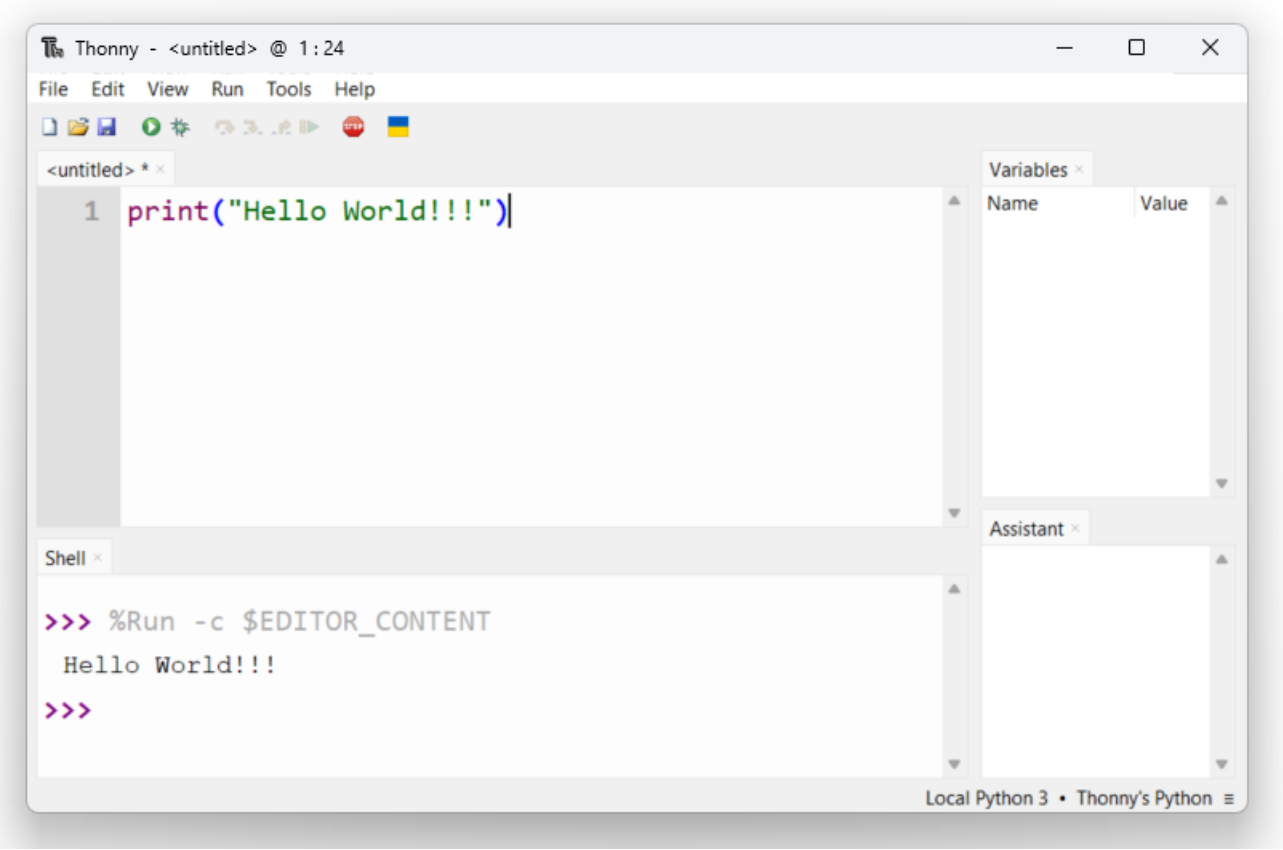
- **Drag or copy/paste .uf2** file to **RPI-RP2 drive**

- PI-RP2 drive will disappear, and **new disk drive CIRCUITPY** will appear



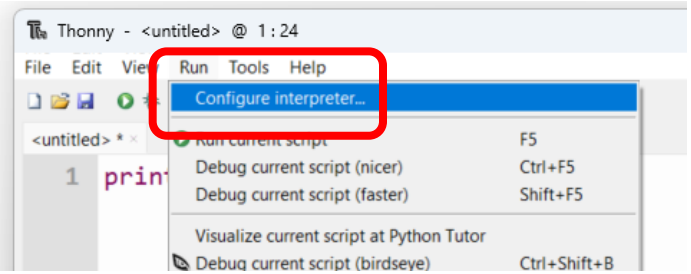
Thonny IDE

Download, install and run. https://thonny.org



Thonny IDE – Setup CircuitPython

- Click **Run** > **Configure interpreter…**



- In **Interpreter Tab**, set:

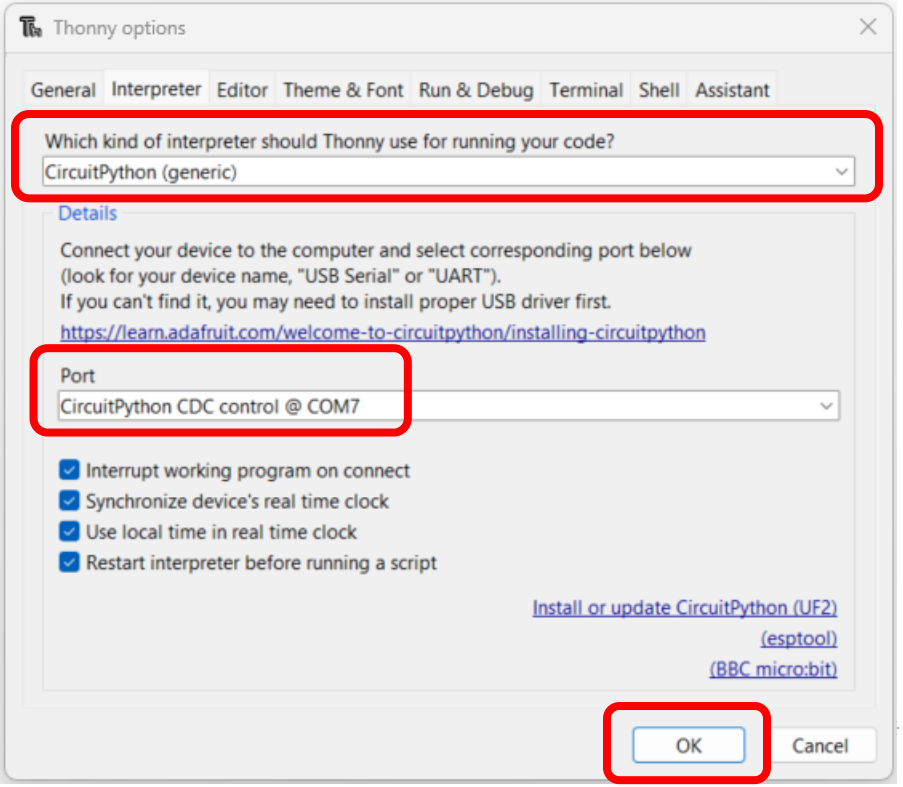
Interpreter:

CircuitPython (generic)

Port:

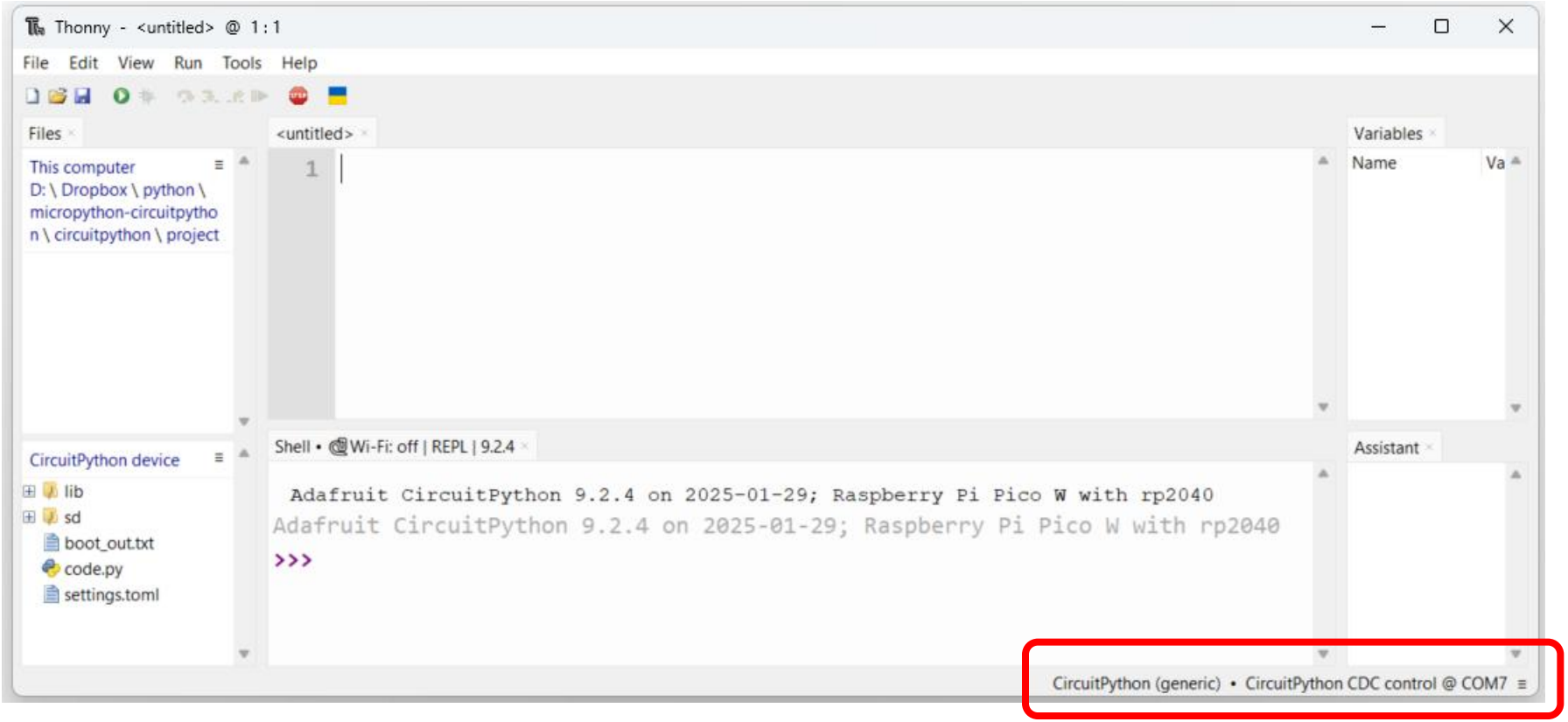
CircuitPython CDC Control

@ COMxx



- Click **OK**

Thonny IDE – Setup CircuitPython (cont)

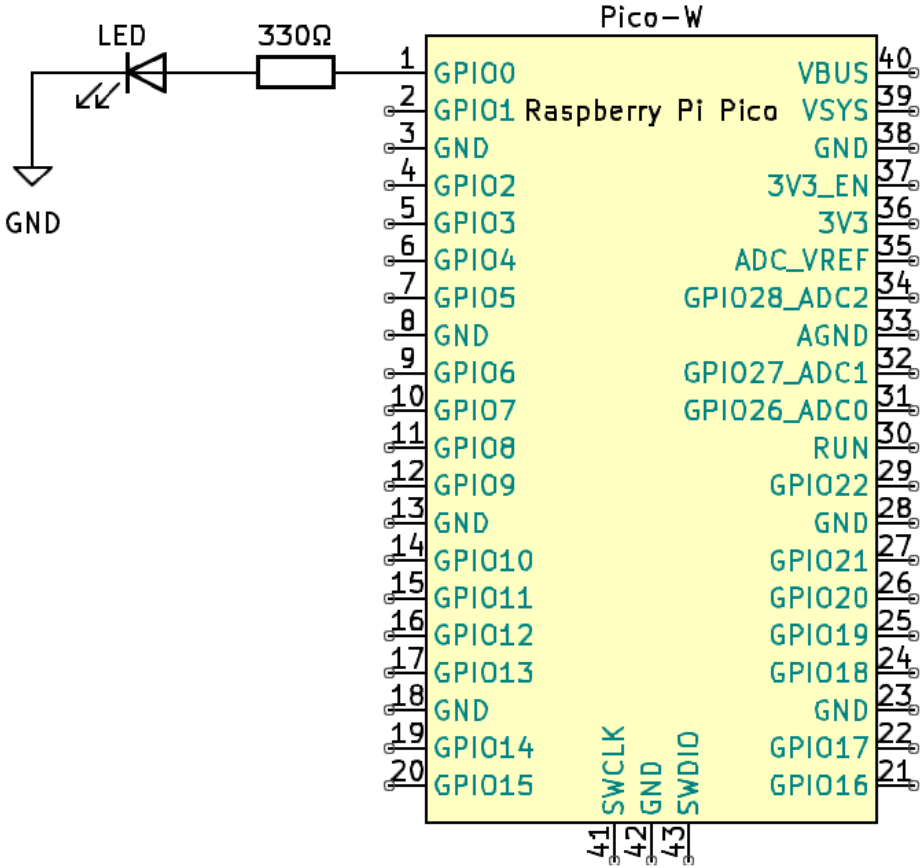


Blink build-in LED

```
import time
import board
from digitalio import DigitalInOut, Direction

led = DigitalInOut(board.LED)
led.direction = Direction.OUTPUT

while True:
    led.value = True
    time.sleep(1)
    led.value = False
    time.sleep(1)
```

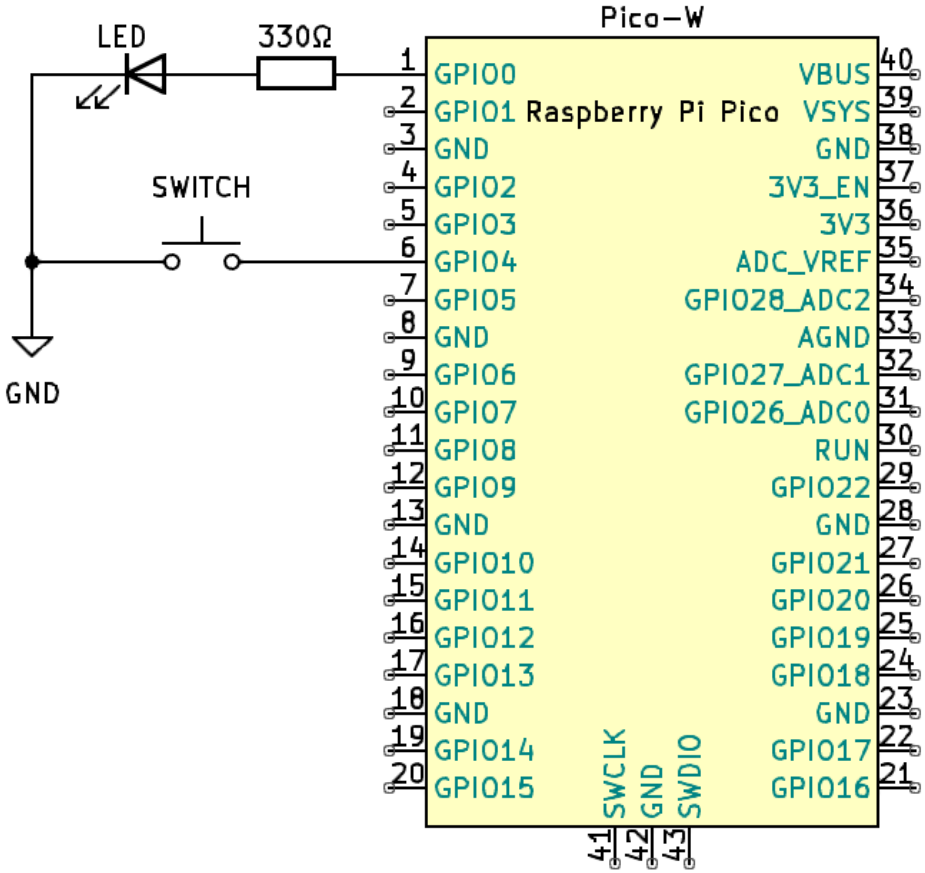
Digital Output

```python
import time
import board
from digitalio import DigitalInOut, Direction

led = DigitalInOut(board.GP0)
led.direction = Direction.OUTPUT

while True:
    led.value = True
    print("ON")
    time.sleep(1)

    led.value = False
    print("OFF")
    time.sleep(1)
```



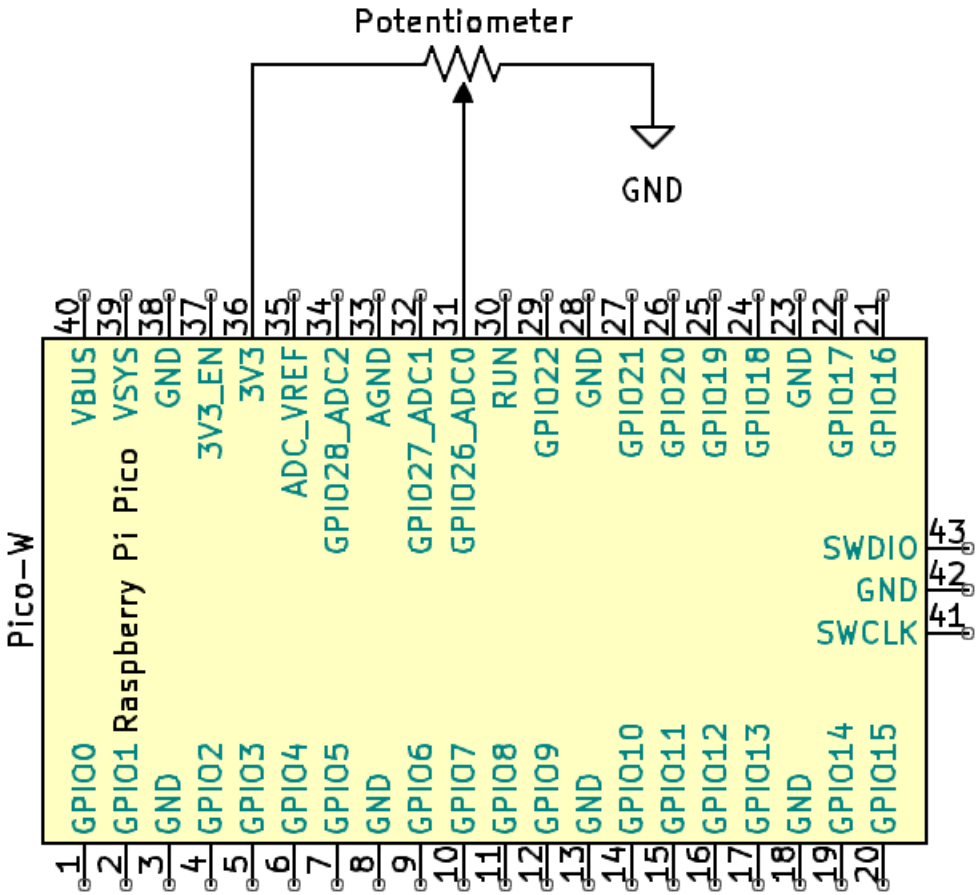
Digital Input

```python
import time
import board
from digitalio import DigitalInOut, Direction, Pull

led = DigitalInOut(board.GP0)
led.direction = Direction.OUTPUT

switch = DigitalInOut(board.GP4)
switch.direction = Direction.INPUT
switch.pull = Pull.UP

while True:
    if switch.value:
        led.value = False
        print("OFF")
    else:
        led.value = True
        print("ON")

    time.sleep(0.01)
```



Analog Input

```python
import time
import board
from analogio import AnalogIn

analog_in = AnalogIn(board.A0)

def get_voltage(value):
    """
    Count up from 0 to 65535
    - 12 bit ADC however micropython
    scale to 16 bit
    """
    return (value * 3.3) / 65536

while True:
    analogIn = analog_in.value
    print(f"adc = {analogIn}")

    voltage = get_voltage(analogIn)
    print(f"volt = {voltage} V")

    time.sleep(0.1)
```

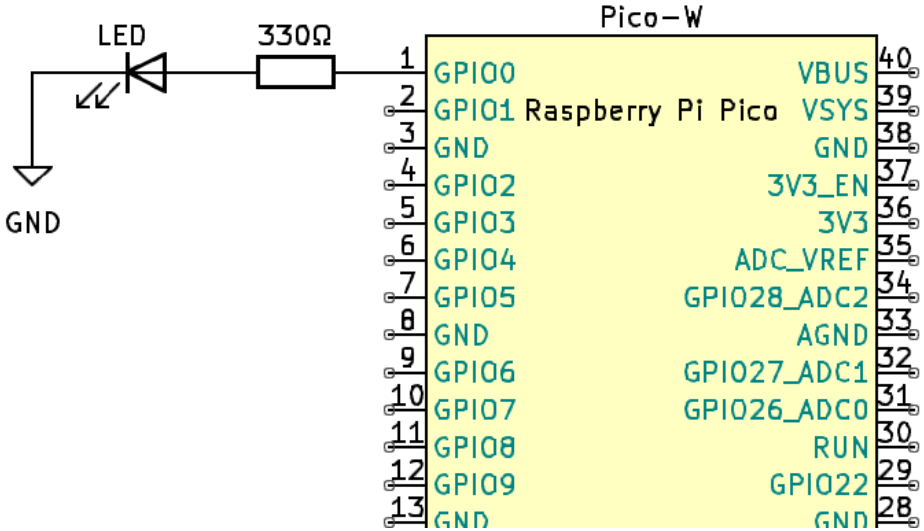


Analog Output – N/A

```
import board
from analogio import AnalogOut

analog_out = AnalogOut(board.A0)

while True:
    # Count up from 0 to 65535
    for i in range(0, 65536):
        analog_out.value = i
```

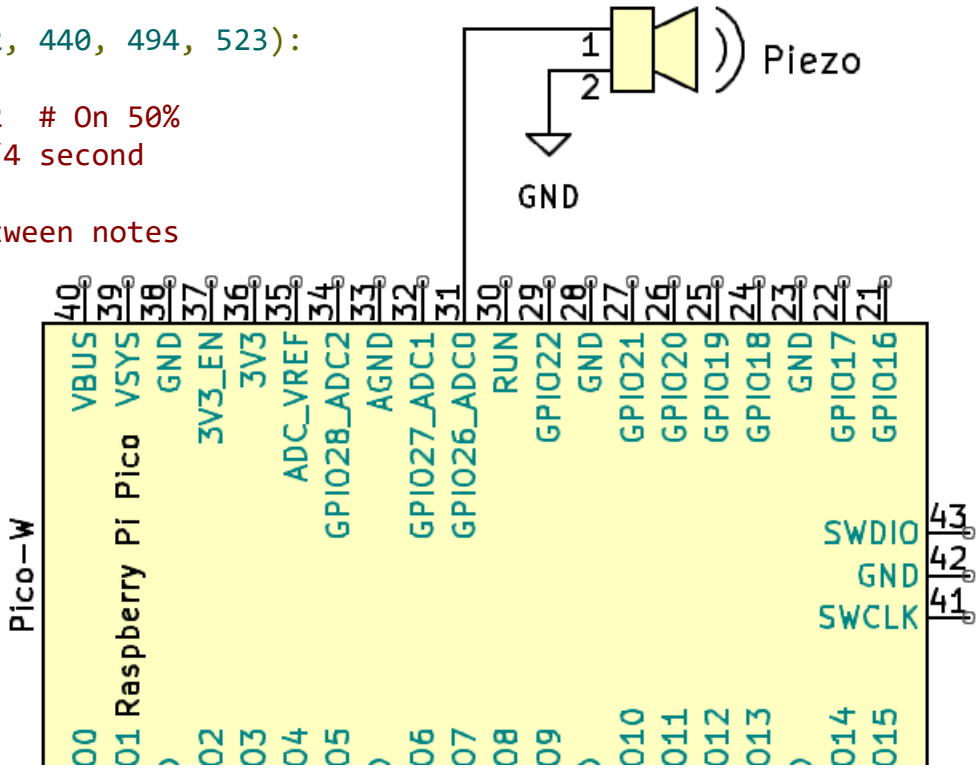
PWM – fixed frequency

```
import board
from pwmio import PWMOut

led = PWMOut(board.GP0, frequency=5000, duty_cycle=0)

while True:
    for i in range(100):
        # PWM LED up and down
        if i < 50:
            led.duty_cycle = int(i * 2 * 65535 / 100)  # Up
        else:
            led.duty_cycle = 65535 - int((i - 50) * 2 * 65535 / 100)  # Down
        time.sleep(0.01)
```



PWM – variable frequency

```
import time
import board
import pwmio

piezo = pwmio.PWMOut(board.A0, duty_cycle=0, frequency=440, variable_frequency=True)

while True:
    for f in (262, 294, 330, 349, 392, 440, 494, 523):
        piezo.frequency = f
        piezo.duty_cycle = 65536 // 2  # On 50%
        time.sleep(0.25)  # On for 1/4 second
        piezo.duty_cycle = 0  # Off
        time.sleep(0.05)  # Pause between notes
    time.sleep(0.5)
```



CircuitPython Libraries

- Contains all current libraries available
- provide additional functionality and support external devices
- Stored on **CIRCUITPY drive** in a folder called **lib**
- Download at https://circuitpython.org/libraries

Bundles

Bundle for Version 9.x

This bundle is built for use with CircuitPython 9.x.x. If you are using CircuitPython 9, please download this bundle. The .mpy format changed as of CircuitPython 9: 8.x libraries are *not* compatible.

adafruit-circuitpython-bundle-9.x-mpy-20250319.zip

https://circuitpython.org/libraries

DHT22

```python
import time
import board
import adafruit_dht

dhtDevice = adafruit_dht.DHT22(board.GP22)

while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        humidity = dhtDevice.humidity
        print(f"Temp: {temperature_c} C    Humidity: {humidity}% ")

    except RuntimeError as error:
        # Errors happen fairly often, DHT's are hard to read, just keep going
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)
```