HackED 2026
Introduction
Raspberry Pi RP2040

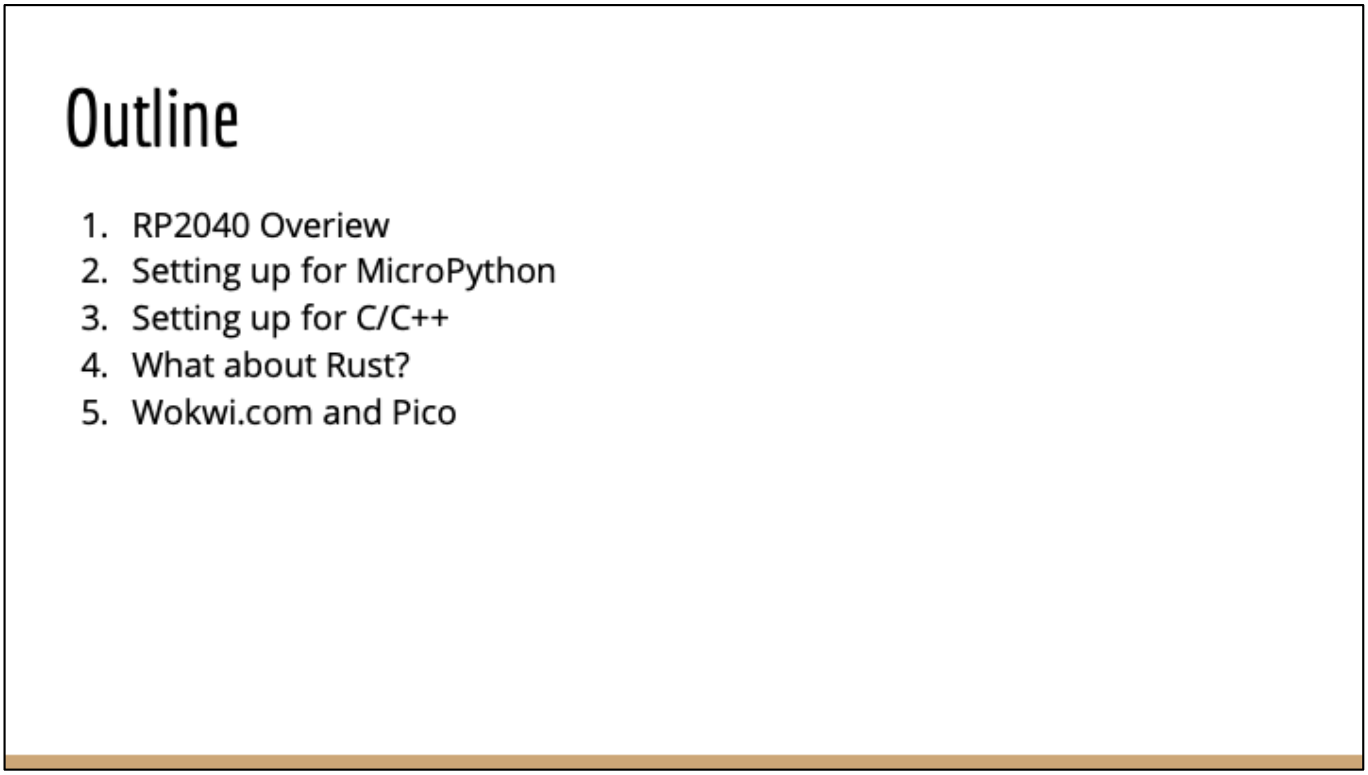
Steven Knudsen, PhD PEng
knud@ualberta.ca

DEPARTMENT OF

Electrical and Computer Engineering

Innovation
Creativity
Entrepreneurship
Incubator

www.iceincubator.com



Outline

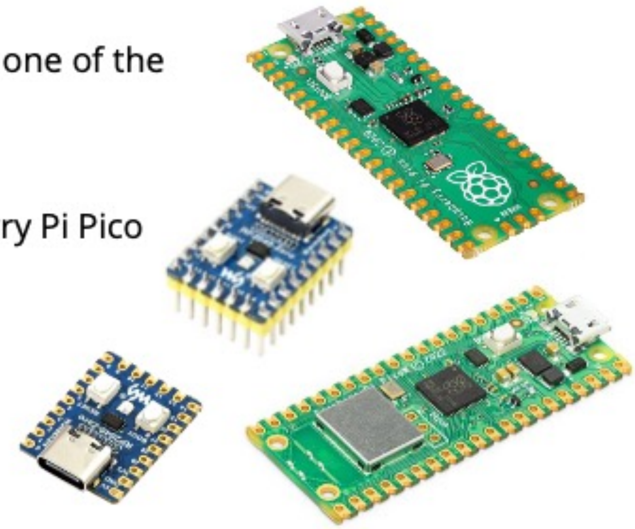
1. RP2040 Overeview
2. Setting up for MicroPython
3. Setting up for C/C++
4. What about Rust?
5. Wokwi.com and Pico

Sources for the talk

github.com/StevenKnudsen/HackED2026Workshop

Assumptions

- You know how to program in at least one of the languages
  - Python or C
- You have an RP2040-Zero or Raspberry Pi Pico



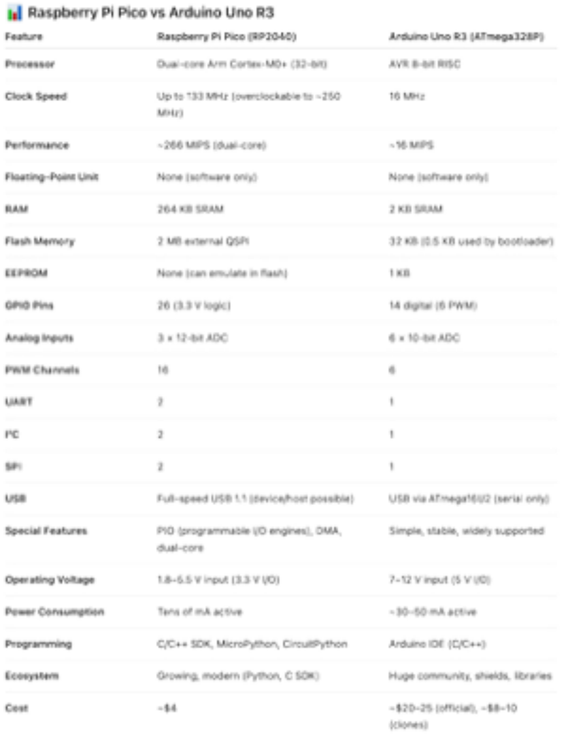
Images from digikey.ca
https://www.waveshare.com/wiki/RP2040-Zero

RP2040 - Raspberry Pi Pico Processor

- The RP2040 is the first processor chip designed and manufactured by (probably for) Raspberry Pi Ltd.
- It’s a 32-bit dual-core ARM Cortex-M0+ microcontroller
- It is very powerful, very inexpensive ($CA1.09 Qty 1)
- The dual-cores are clocked at 133 MHz and can be overclocked to 250 MHz
- This means under ideal conditions we can get from 266 to 500 million instructions per second (MIPS)

Shameless Plug

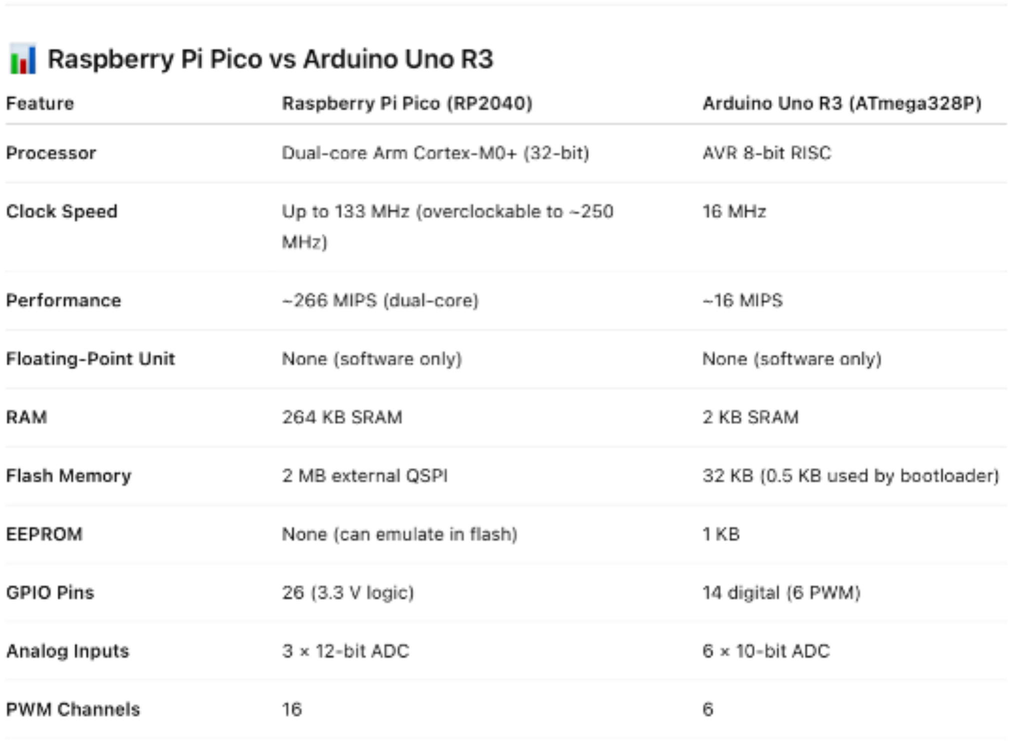
- In Mechatronics we have selected to base many labs and courses on using the Raspberry Pi Pico
- Why?
- It is contemporary
- Exceptionally capable
- Exceptional value
- Something you might use in a Mechatronics system



📊 Raspberry Pi Pico vs Arduino Uno R3

Feature	Raspberry Pi Pico (RP2040)	Arduino Uno R3 (ATmega328P)
Processor	Dual-core Arm Cortex-M0+ (32-bit)	AVR 8-bit RISC
Clock Speed	Up to 133 MHz (overclockable to ~250 MHz)	16 MHz
Performance	~266 MIPS (dual-core)	~16 MIPS
Floating-Point Unit	None (software only)	None (software only)
RAM	264 KB SRAM	2 KB SRAM
Flash Memory	2 MB external QSPI	32 KB (0.5 KB used by bootloader)
EEPROM	None (can emulate in flash)	1 KB
GPIO Pins	26 (3.3 V logic)	14 digital (6 PWM)
Analog Inputs	3 × 12-bit ADC	6 × 10-bit ADC
PWM Channels	16	6
UART	2	1
I^2C	2	1
SPI	2	1
USB	Full-speed USB 1.1 (device/host possible)	USB via ATmega16U2 (serial only)
Special Features	PIO (programmable I/O engines), DMA, dual-core	Simple, stable, widely supported
Operating Voltage	1.8–5.5 V input (3.3 V I/O)	7–12 V input (5 V I/O)
Power Consumption	Tens of mA active	~30–50 mA active
Programming	C/C++ SDK, MicroPython, CircuitPython	Arduino IDE (C/C++)
Ecosystem	Growing, modern (Python, C SDK)	Huge community, shields, libraries
Cost	~$4	~$20–25 (official), ~$8–10 (clones)

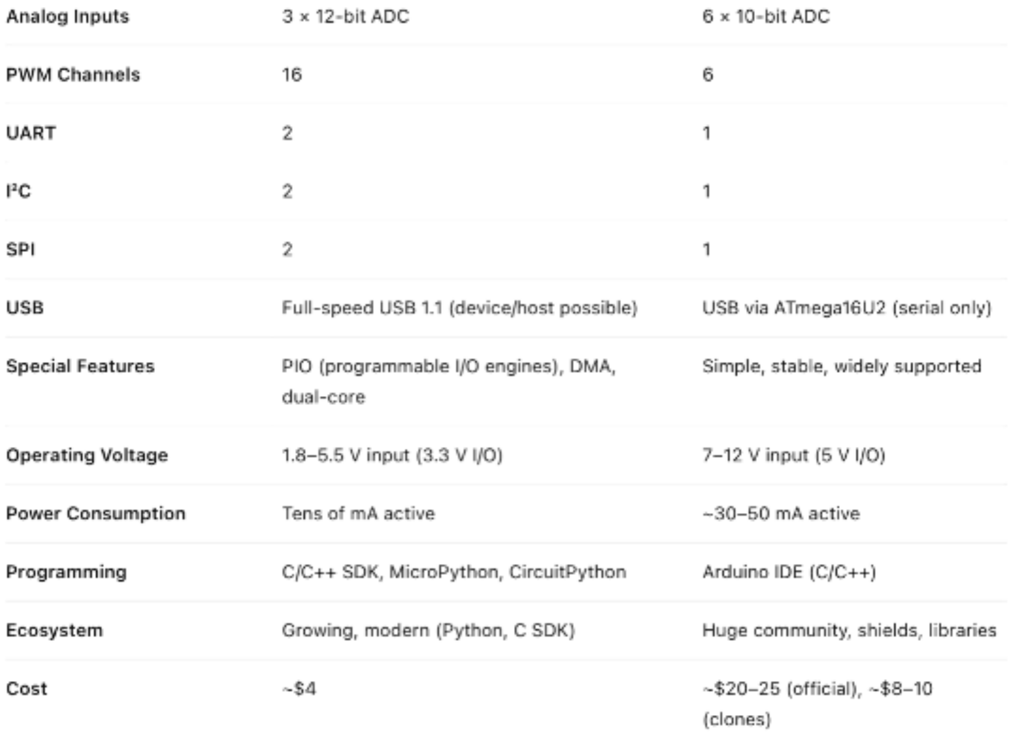
RP2040 vs Arduino



📊 Raspberry Pi Pico vs Arduino Uno R3

Feature	Raspberry Pi Pico (RP2040)	Arduino Uno R3 (ATmega328P)
Processor	Dual-core Arm Cortex-M0+ (32-bit)	AVR 8-bit RISC
Clock Speed	Up to 133 MHz (overclockable to ~250 MHz)	16 MHz
Performance	~266 MIPS (dual-core)	~16 MIPS
Floating-Point Unit	None (software only)	None (software only)
RAM	264 KB SRAM	2 KB SRAM
Flash Memory	2 MB external QSPI	32 KB (0.5 KB used by bootloader)
EEPROM	None (can emulate in flash)	1 KB
GPIO Pins	26 (3.3 V logic)	14 digital (6 PWM)
Analog Inputs	3 × 12-bit ADC	6 × 10-bit ADC
PWM Channels	16	6

RP2040 vs Arduino



Analog Inputs	3 × 12-bit ADC	6 × 10-bit ADC
PWM Channels	16	6
UART	2	1
I²C	2	1
SPI	2	1
USB	Full-speed USB 1.1 (device/host possible)	USB via ATmega16U2 (serial only)
Special Features	PIO (programmable I/O engines), DMA, dual-core	Simple, stable, widely supported
Operating Voltage	1.8–5.5 V input (3.3 V I/O)	7–12 V input (5 V I/O)
Power Consumption	Tens of mA active	~30–50 mA active
Programming	C/C++ SDK, MicroPython, CircuitPython	Arduino IDE (C/C++)
Ecosystem	Growing, modern (Python, C SDK)	Huge community, shields, libraries
Cost	~$4	~$20–25 (official), ~$8–10 (clones)

Raspberry Pi Pico and Pico W

Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264 kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-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 I/O (PIO) state machines for custom peripheral support

Images from digikey.ca

Raspberry Pi Pico and Pico W

$CA5.80 at Mouser.ca

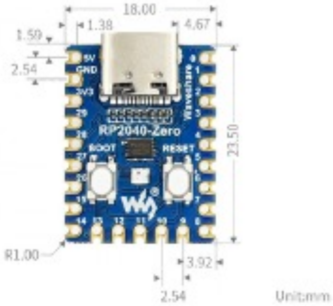
$CA8.34 at canada.newark.com

Images from digikey.ca

RP2040-Zero

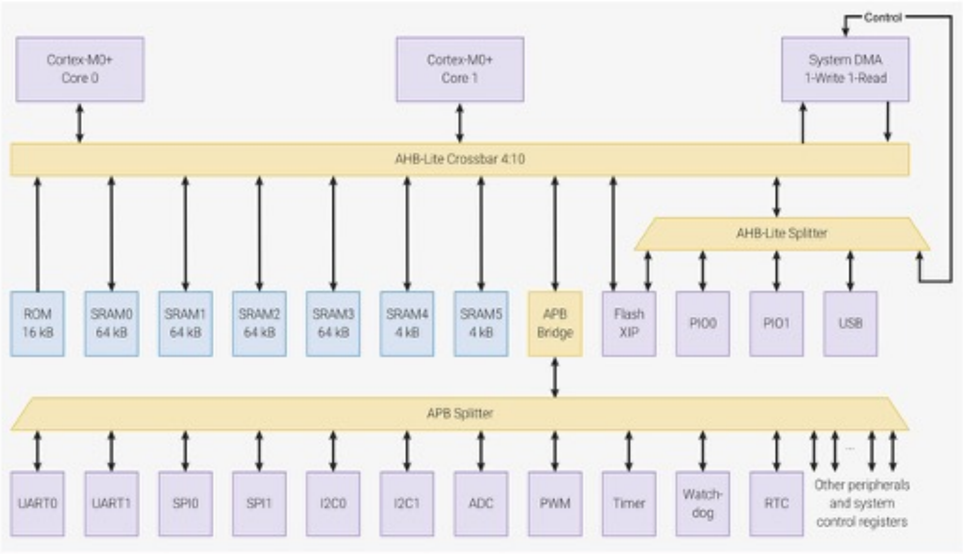
Same processor, a little different form factor

- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Type-C connector; more contemporary, easier to use 26 × multi-function GPIO pins
- Castellated module allows soldering directly to carrier boards
- 29 GPIO pins of RP2040 (20 can be led out through pin headers, the rest can be acccessed by soldering)



RP2040** Architecture

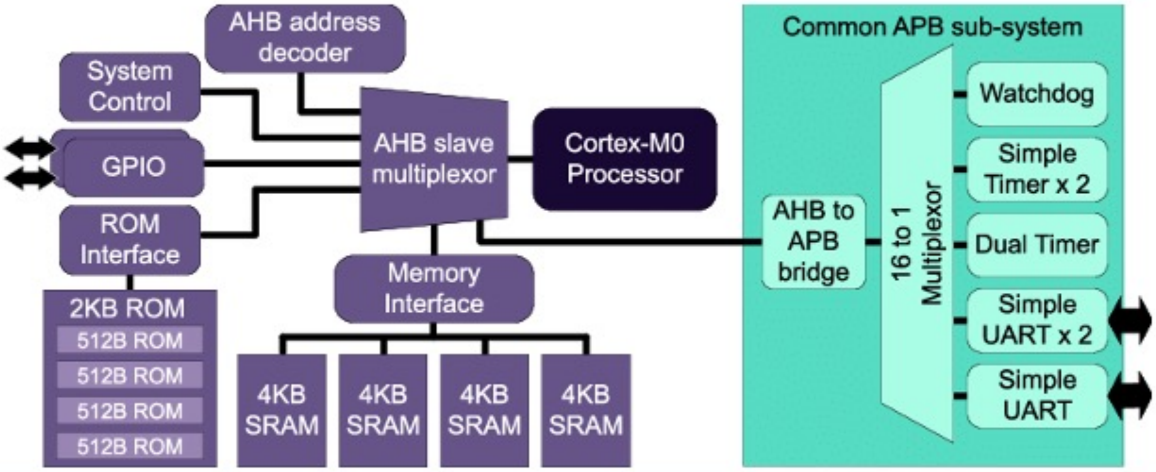
- Advanced high-performance bus matrix (crossbar)
- Lots of memory (ROM, SRAM, FLASH)
- Lots of peripherals, communications, ADC, Timer, real-time clock (RTC)



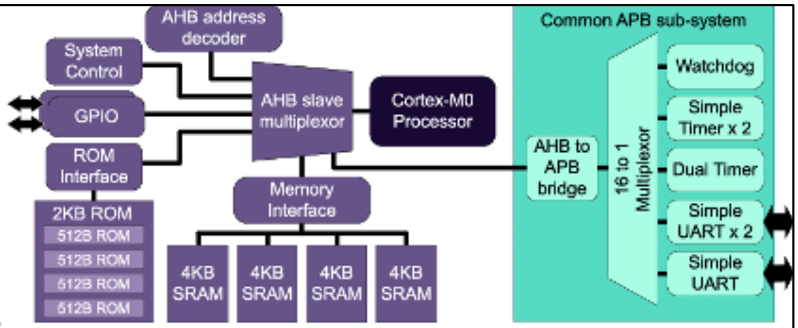
** Main reference for the RP2040 material is the RP2040 Datasheet, 2025-02-20

Basic Architecture

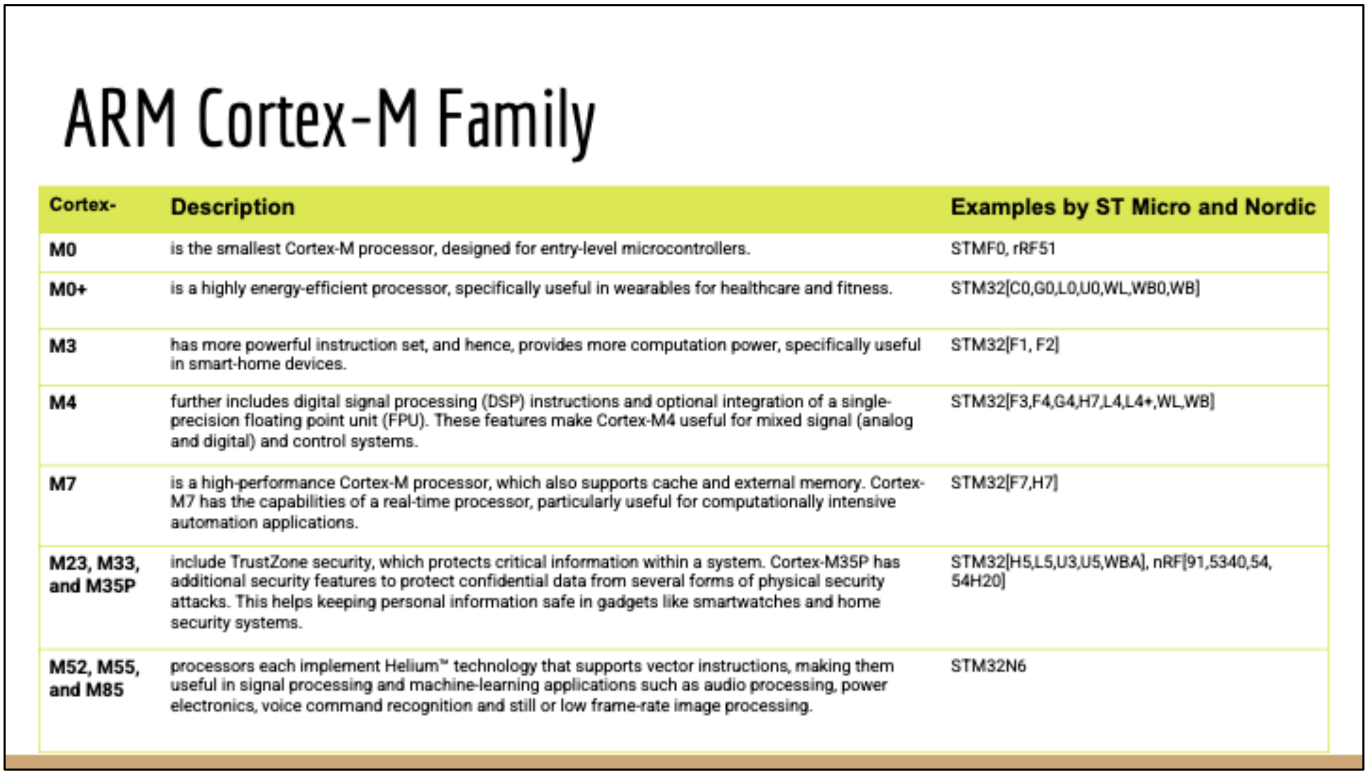
- Advanced high-performance bus (AHB) matrix



Basic Architecture



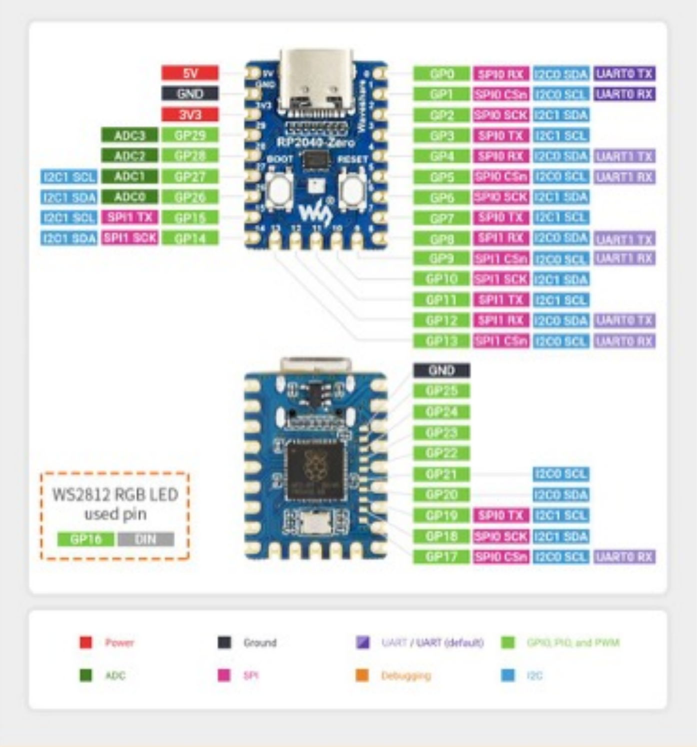
- Advanced Peripheral Bus (APB)
- Read-only Memory (ROM)
- Manufacturer-specific program and data to support the processor's operations, e.g,. specialized embedded devices like analogue-to-digital converters
- GPIO – General Purpose Input/Output
- Typically control logic levels (signal) on external pins
- SRAM – Static Random Access Memory
- low-power RAM with fast access
- UART – Universal Asynchronous Receiver-Transmitter
- Supports serial data communications (send bits on one wire, receive on another)
- Timer – Can generate events based on programmed time intervals
- Watchdog – Used to monitor the processor or peripherals to make sure they are alive



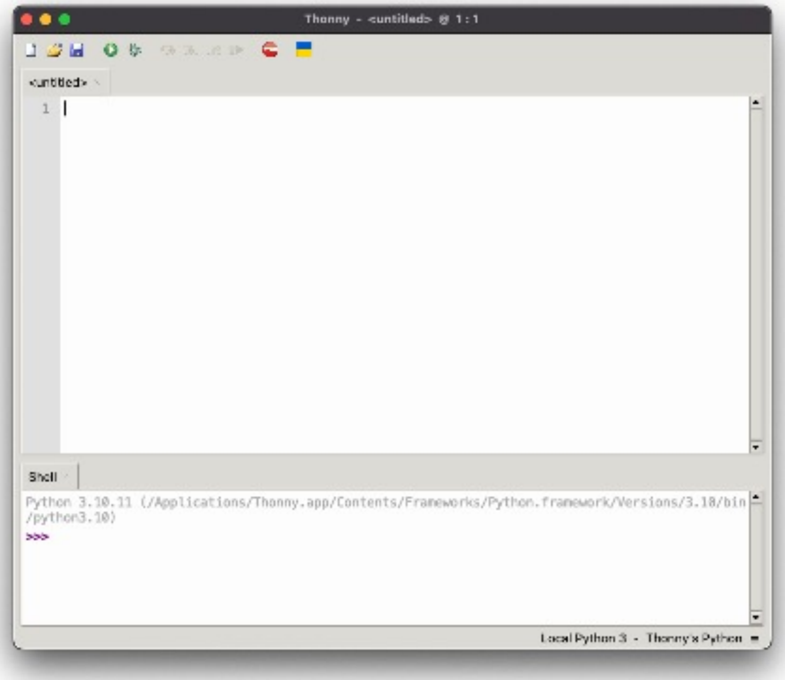
ARM Cortex-M Family

Cortex-	Description	Examples by ST Micro and Nordic
M0	is the smallest Cortex-M processor, designed for entry-level microcontrollers.	STMF0, rRF51
M0+	is a highly energy-efficient processor, specifically useful in wearables for healthcare and fitness.	STM32[C0,G0,L0,U0,WL,WB0,WB]
M3	has more powerful instruction set, and hence, provides more computation power, specifically useful in smart-home devices.	STM32[F1, F2]
M4	further includes digital signal processing (DSP) instructions and optional integration of a single-precision floating point unit (FPU). These features make Cortex-M4 useful for mixed signal (analog and digital) and control systems.	STM32[F3,F4,G4,H7,L4,L4+,WL,WB]
M7	is a high-performance Cortex-M processor, which also supports cache and external memory. Cortex-M7 has the capabilities of a real-time processor, particularly useful for computationally intensive automation applications.	STM32[F7,H7]
M23, M33, and M35P	include TrustZone security, which protects critical information within a system. Cortex-M35P has additional security features to protect confidential data from several forms of physical security attacks. This helps keeping personal information safe in gadgets like smartwatches and home security systems.	STM32[H5,L5,U3,U5,WBA], nRF[91,5340,54, 54H20]
M52, M55, and M85	processors each implement Helium™ technology that supports vector instructions, making them useful in signal processing and machine-learning applications such as audio processing, power electronics, voice command recognition and still or low frame-rate image processing.	STM32N6

Rp2040-Zero


5V
GND
3V3
ADC3 GP29
ADC2 GP28
I2C1 SCL ADC1 GP27
I2C1 SDA ADC0 GP26
I2C1 SCL SPI1 TX GP15
I2C1 SDA SPI1 SCK GP14
RP2040-Zero
BOOT
RESET
Waveshare
GP0 SPI0 RX I2C0 SDA UART0 TX
GP1 SPI0 CSn I2C0 SCL UART0 RX
GP2 SPI0 SCK I2C1 SDA
GP3 SPI0 TX I2C1 SCL
GP4 SPI0 RX I2C0 SDA UART1 TX
GP5 SPI0 CSn I2C0 SCL UART1 RX
GP6 SPI0 SCK I2C1 SDA
GP7 SPI0 TX I2C1 SCL
GP8 SPI1 RX I2C0 SDA UART1 TX
GP9 SPI1 CSn I2C0 SCL UART1 RX
GP10 SPI1 SCK I2C1 SDA
GP11 SPI1 TX I2C1 SCL
GP12 SPI1 RX I2C0 SDA UART0 TX
GP13 SPI1 CSn I2C0 SCL UART0 RX
GND
GP25
GP24
GP23
GP22
GP21 I2C0 SCL
GP20 I2C0 SDA
GP19 SPI0 TX I2C1 SCL
GP18 SPI0 SCK I2C1 SDA
GP17 SPI0 CSn I2C0 SCL UART0 RX
WS2812 RGB LED used pin
GP16 DIN
🟥 Power
⬛ Ground
🟪 UART / UART (default)
🟩 GPIO, PIO, and PWM
🟩 ADC
🟪 SPI
🟧 Debugging
🟦 I2C

Install Thonny (Win/Lin/macOS)



```
Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin
/python3.10)
>>>
```

This is Thonny, a beginner Python IDE. Happens to be running on a MacBook Pro, but same for other OSes

Zero in boot select (bootsel) mode

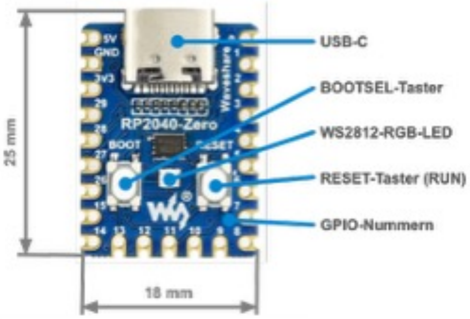
Plug Zero into USB

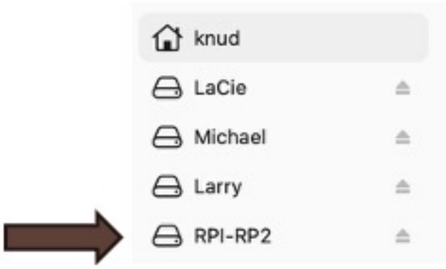
Press and hold Boot (select) button

Press and release Reset button

Release Boot button

Open Thonny



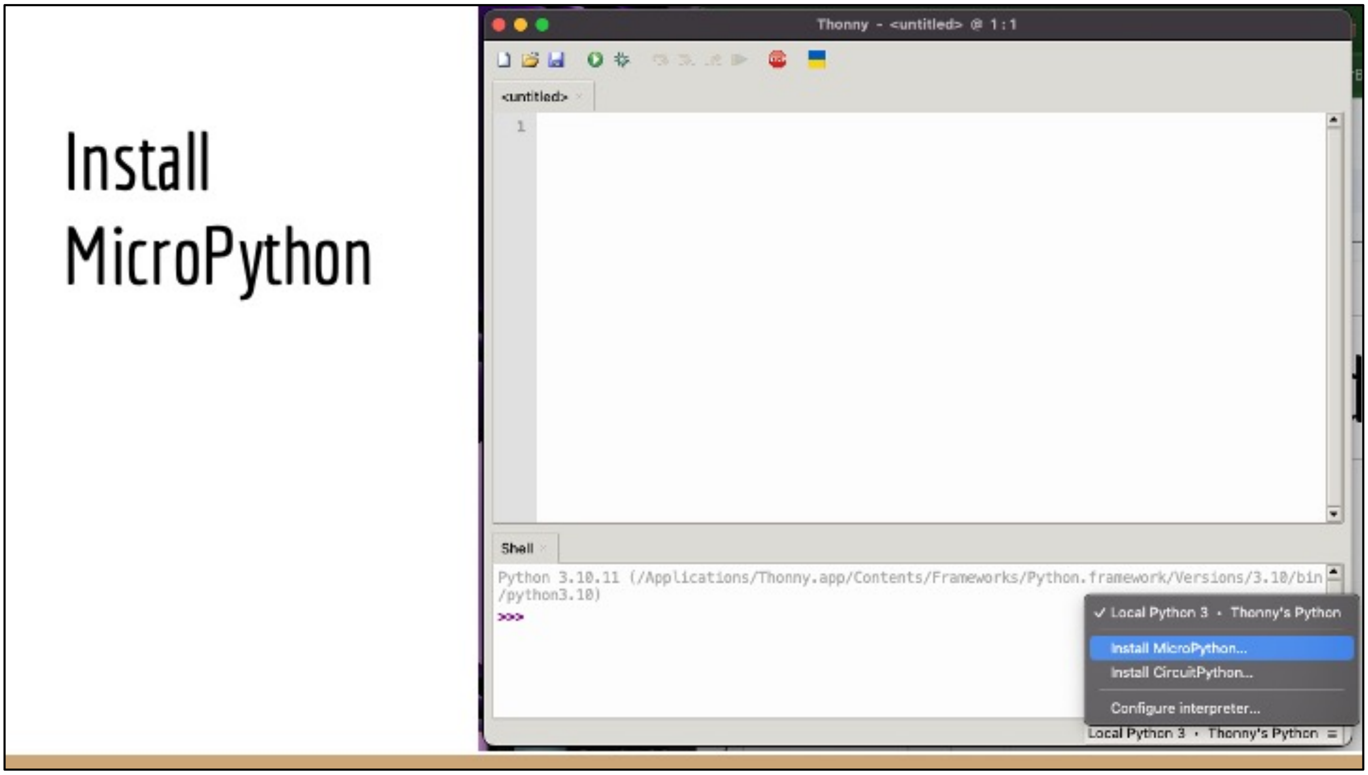

knud

LaCie

Michael

Larry

RPI-RP2



Install MicroPython

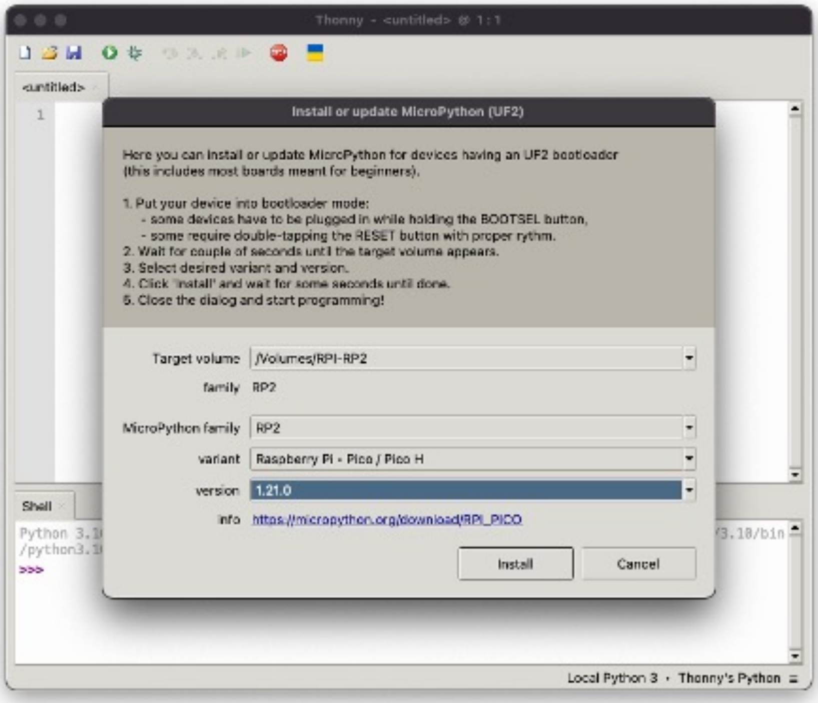
Thonny - <untitled> @ 1 : 1
<untitled>
Shell

```
Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin
/python3.10)
>>>
```

✓ Local Python 3 • Thonny's Python
Install MicroPython...
Install CircuitPython...
Configure interpreter...
Local Python 3 • Thonny's Python ≡

Open the hamburger menu in the lower right corner and select install micropython

Install dialogue



Install or update MicroPython (UF2)

Here you can install or update MicroPython for devices having an UF2 bootloader
(this includes most boards meant for beginners).

1. Put your device into bootloader mode:
   - some devices have to be plugged in while holding the BOOTSEL button,
   - some require double-tapping the RESET button with proper rythm.
2. Wait for couple of seconds until the target volume appears.
3. Select desired variant and version.
4. Click 'Install' and wait for some seconds until done.
5. Close the dialog and start programming!

Target volume: /Volumes/RPI-RP2

family: RP2

MicroPython family: RP2

variant: Raspberry Pi • Pico / Pico H

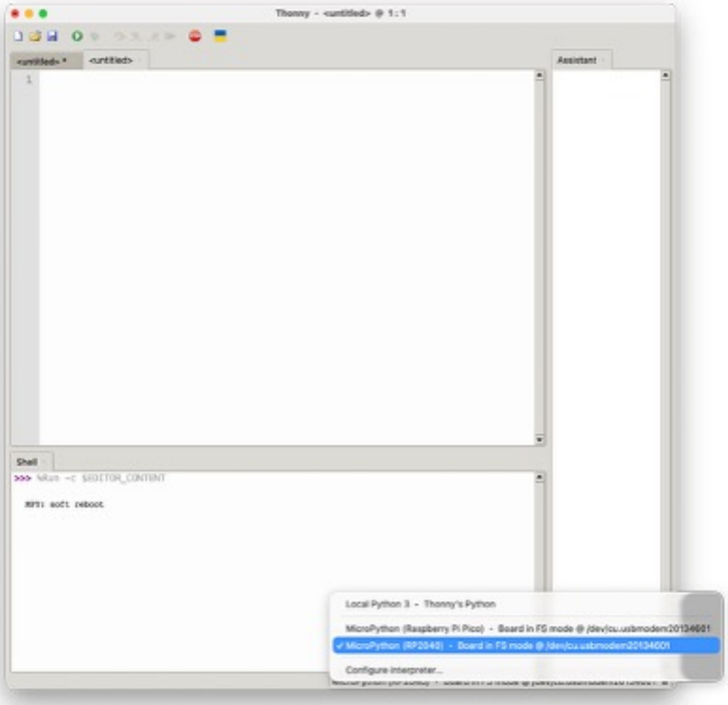
version: 1.21.0

info: https://micropython.org/download/RPI_PICO

```
Python 3.1
/python3.1
>>>
```

The RPi will disconnect when this is done, then reconnect

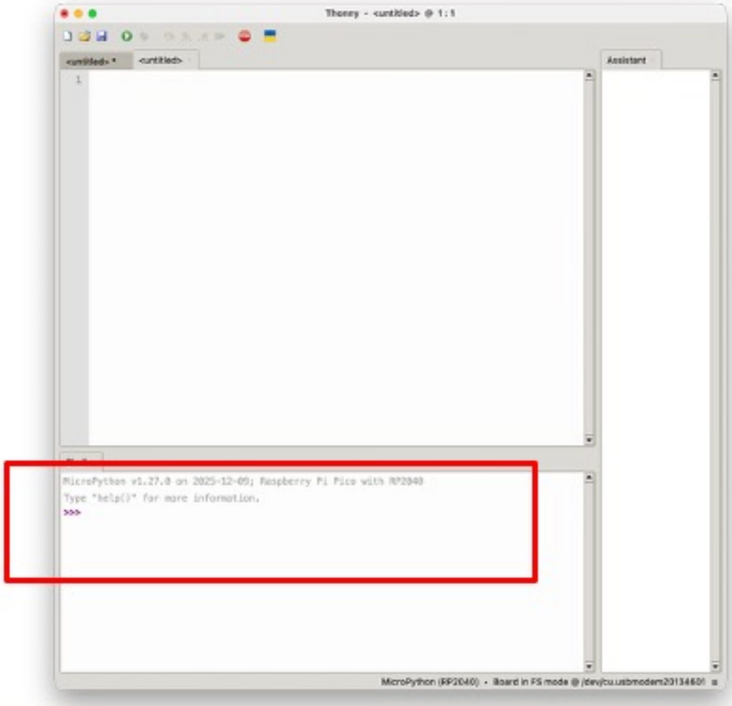
Connect


Thonny - <untitled> @ 1:1
<untitled> *
<untitled>
Assistant
Shell
Local Python 3 - Thonny's Python
MicroPython (Raspberry Pi Pico) - Board in FS mode @ /dev/cu.usbmodem20134601
✓ MicroPython (RP2040) - Board in FS mode @ /dev/cu.usbmodem20134601
Configure interpreter...

```
>>> %Run -c $EDITOR_CONTENT

  MPY: soft reboot
```

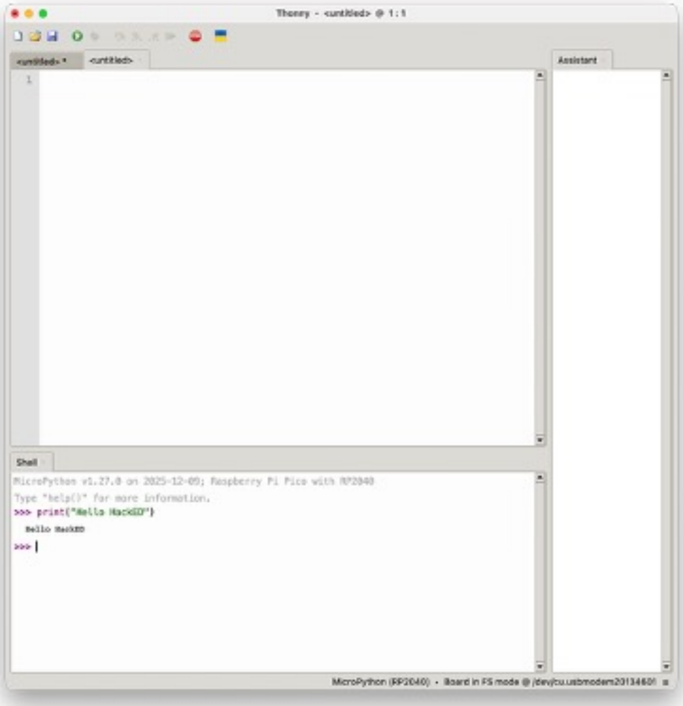
We need to connect to the Pico. Again, open the hamburger menu. Notice these are both really the same, so it doesn't matter which we choose.



Thonny - <untitled> @ 1 : 1
<untitled> *
<untitled>
Assistant
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2013460!

```
MicroPython v1.27.0 on 2025-12-09; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

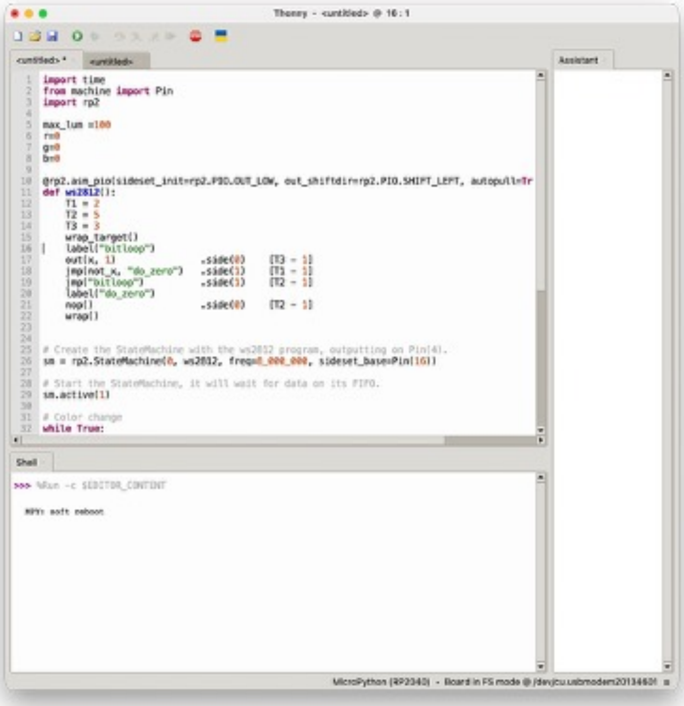
Success!



```
MicroPython v1.27.0 on 2025-12-09; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello HackED")
  Hello HackED
>>> 
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem20134601

Cycle the RGB in Python



```
import time
from machine import Pin
import rp2

max_lum =100
r=0
g=0
b=0

@rp2.asm_pio(sideset_init=rp2.PIO.OUT_LOW, out_shiftdir=rp2.PIO.SHIFT_LEFT, autopull=Tr
def ws2812():
    T1 = 2
    T2 = 5
    T3 = 3
    wrap_target()
    label("bitloop")
    out(x, 1)               .side(0)    [T3 - 1]
    jmp(not_x, "do_zero")   .side(1)    [T1 - 1]
    jmp("bitloop")          .side(1)    [T2 - 1]
    label("do_zero")
    nop()                   .side(0)    [T2 - 1]
    wrap()


# Create the StateMachine with the ws2812 program, outputting on Pin(4).
sm = rp2.StateMachine(0, ws2812, freq=8_000_000, sideset_base=Pin(16))

# Start the StateMachine, it will wait for data on its FIFO.
sm.active(1)

# Color change
while True:
```

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem2013460!

Live demo

GPIOs

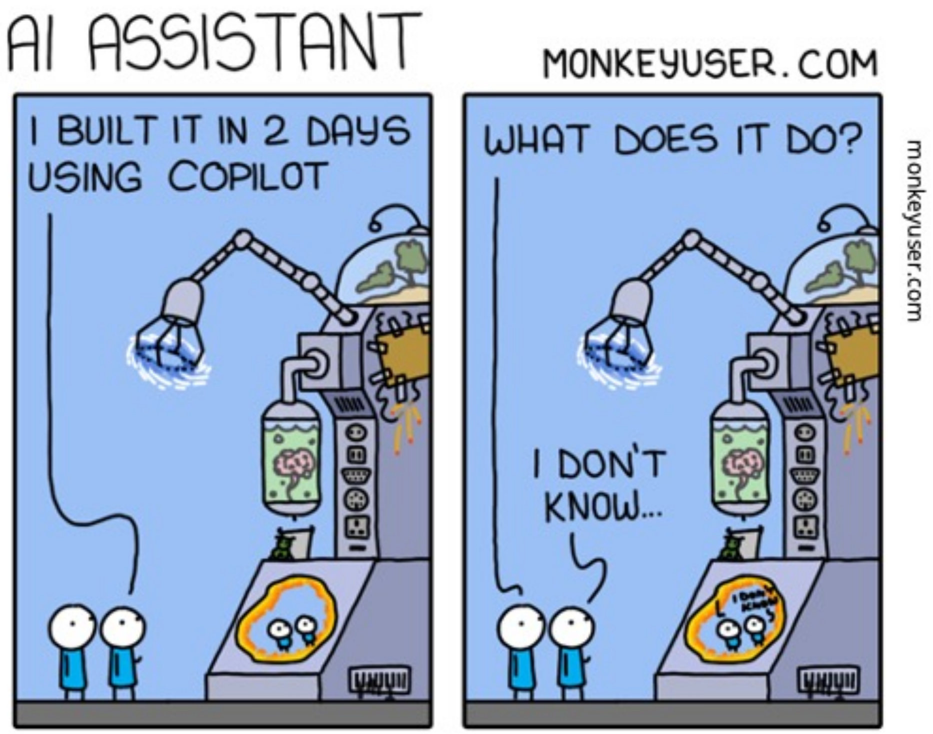
Lots of support for GPIO and other peripherals at MicroPython.org

Quick Reference for MicroPython is found at

https://docs.micropython.org/en/latest/rp2/quickref.html

Support for

- HW Timers, GPIO, UART, PWM, ADC, SW SPI, HW SPI, SW I2C, HW I2C, I2S, RTC, WDT, OneWire, NeoPixel, APA106
- … but probably not …


AI ASSISTANT
MONKEYUSER.COM
I BUILT IT IN 2 DAYS USING COPILOT
WHAT DOES IT DO?
I DON'T KNOW...
I DON'T KNOW
monkeyuser.com

Getting started with the Zero - C/C++

The main reference is the RPi documentation

https://www.raspberrypi.com/documentation/microcontrollers/c_sdk.html

As per that page there are on Github an SDK and Examples repos

Will use Ubuntu 24.04 for the following, but instructions for Win and macOS are on the link above

(Could set up development on a Raspberry Pi, but we won't for HackEDxx)

Toolchain and SDK install

Need `cmake` and the GCC ARM toolchain

```
sudo apt install cmake gcc-arm-none-eabi libnewlib-arm-none-eabi
libstdc++-arm-none-eabi-newlib
```

Make a top-level directory in my `Development` directory to clone the SDK into

```
cd ~Development
mkdir RPi; cd RPi
git clone https://github.com/raspberrypi/pico-sdk.git
```

Clone the HackED2026Workshop examples

```
git clone https://github.com/StevenKnudsen/HackED2026Workshop.git
```

RGB example

Let’s make sure the blink example works

```
cd ~Development/HackED2026Workshop/C
mkdir build; cd build
cmake ..
make -j4
```

The build will take a long time since all the examples will be compiled…

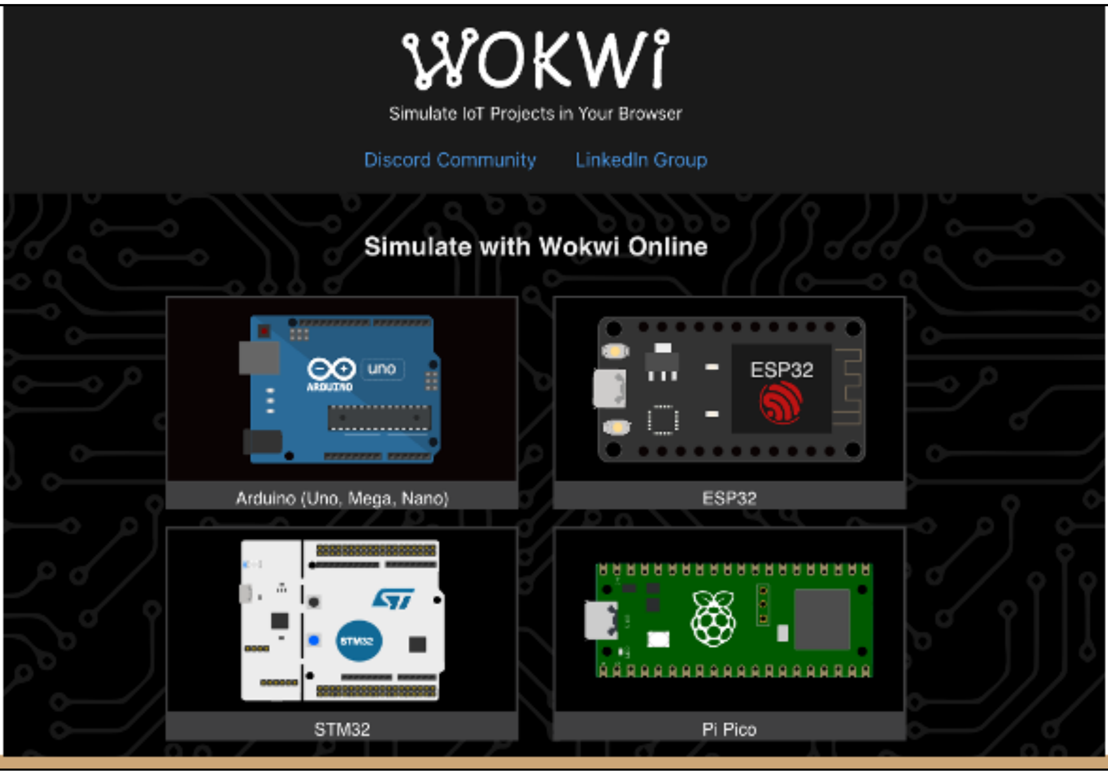
RGB example

When complete, reset the Zero while holding the boot select button. The Pico will mount as a storage device. Drag and drop `~Development/ HackED2026Workshop/C/build/ RP2040_Zero_Test.uf2` onto the Zero

- Will be listed as RPI-RP2 memory device
- Drag and drop the uf2 file

The RGB LED should cycle

One more tool – Wokwi.com



WOKWI

Simulate IoT Projects in Your Browser

Simulate with Wokwi Online

What is Wokwi.com?

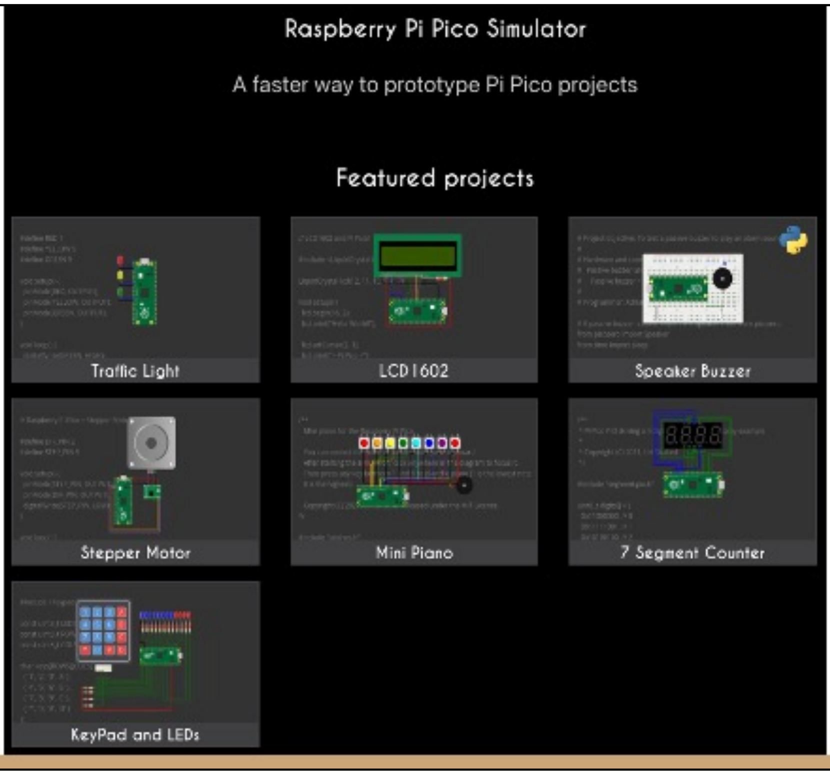
(from docs.wokwi.com)

Wokwi is an online Electronics simulator. You can use it to simulate Arduino, ESP32, STM32, and many other popular boards, parts and sensors.

Here are some quick examples of things you can make with Wokwi:

- Arduino Uno "Hello World"
- Blink an LED on ESP32
- Monitor the weather on ATtiny85
- Control 32 Servos with Arduino Mega
- Animate an LED Matrix with FastLED
- 7 Segment Counter with MicroPython on ESP32

...and Pico!

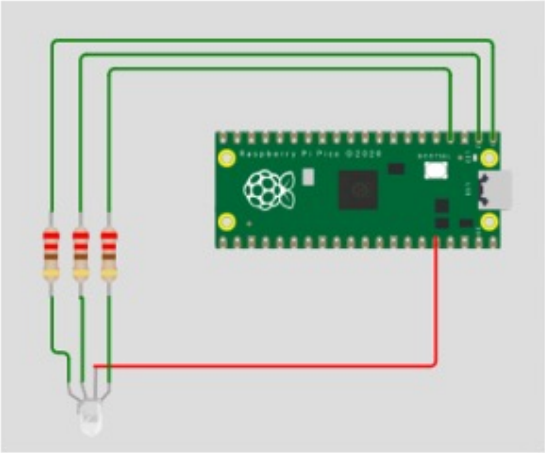


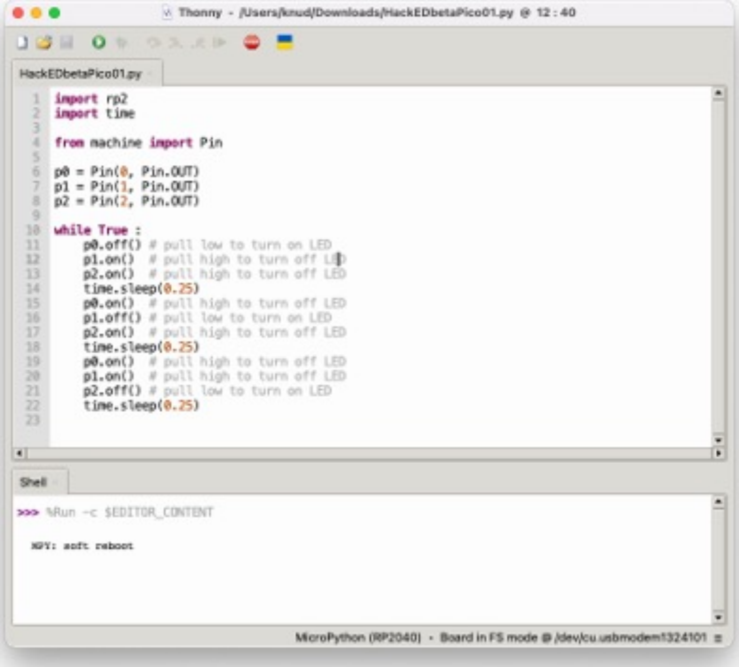
Raspberry Pi Pico Simulator

A faster way to prototype Pi Pico projects

Featured projects

Simple RGB blinky





Thonny - /Users/knud/Downloads/HackEDbetaPico01.py @ 12 : 40
HackEDbetaPico01.py
Shell

```
import rp2
import time

from machine import Pin

p0 = Pin(0, Pin.OUT)
p1 = Pin(1, Pin.OUT)
p2 = Pin(2, Pin.OUT)

while True :
    p0.off() # pull low to turn on LED
    p1.on()  # pull high to turn off LED
    p2.on()  # pull high to turn off LED
    time.sleep(0.25)
    p0.on()  # pull high to turn off LED
    p1.off() # pull low to turn on LED
    p2.on()  # pull high to turn off LED
    time.sleep(0.25)
    p0.on()  # pull high to turn off LED
    p1.on()  # pull high to turn off LED
    p2.off() # pull low to turn on LED
    time.sleep(0.25)
```

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324101

https://wokwi.com/projects/381422162724423681