Getting Started with MicroPython on the Raspberry Pi Pico

Toronto Raspberry Pi Meetup Group
2021-02-11

Stewart Russell – scruss@scruss.com

What, *another* one?

- Not a Linux machine:
  a microcontroller
- Custom silicon, designed by Raspberry Pi Foundation
- Lots of I/O
- Great documentation
- $5.25 CAD, any qty
- Arduino killer

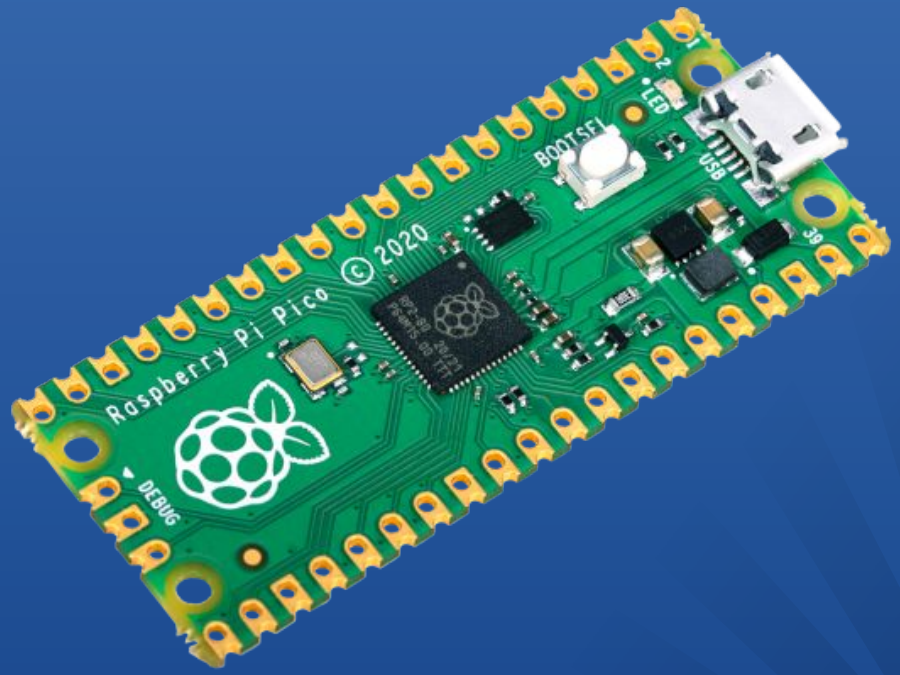


Image credit: Raspberry Pi Foundation

RP2040 Overview

- Dual-core ARM Cortex-M0+ at 133 MHz
- 264 KB RAM
- No Flash storage
  (Pico has 2 MB external)
- 26 × multi-function GPIO pins
- 2 × SPI, 2 × I2C, 2 × UART, 3 × 12-bit ADC, 16 × PWM
- 8 × PIO state machines

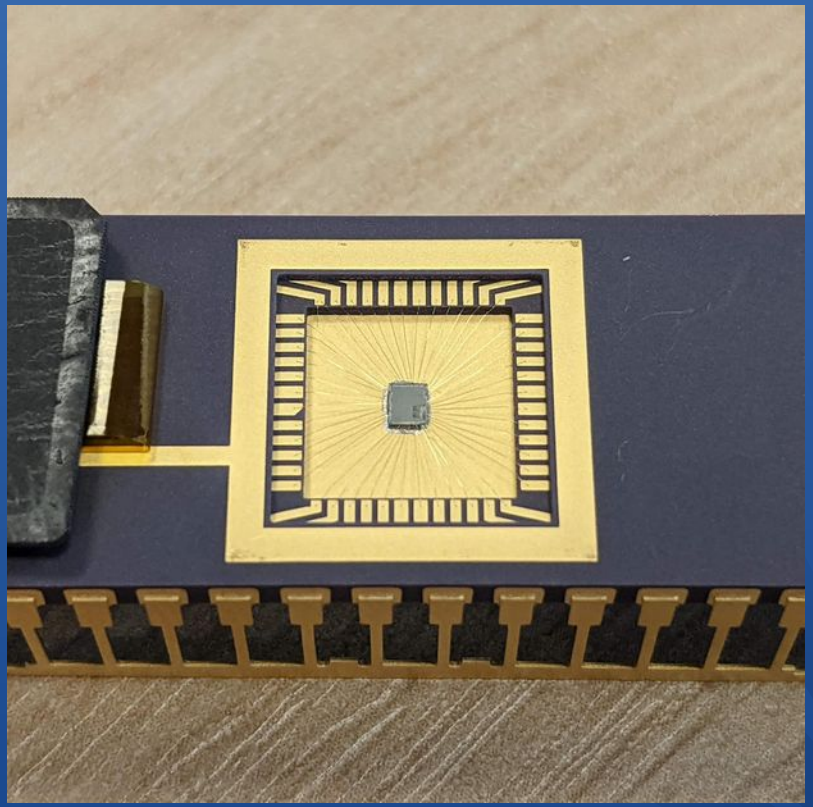


Image credit: Raspberry Pi Foundation

So where do I get one?

- In theory, you can buy as many as you want, but:
  - **BuyaPi**: Sold Out
  - **Canakit**: Sold Out
    [preorders ship Feb 28]
  - **Elmwood**: Sold Out
  - **Newark**, **Digikey**: on order

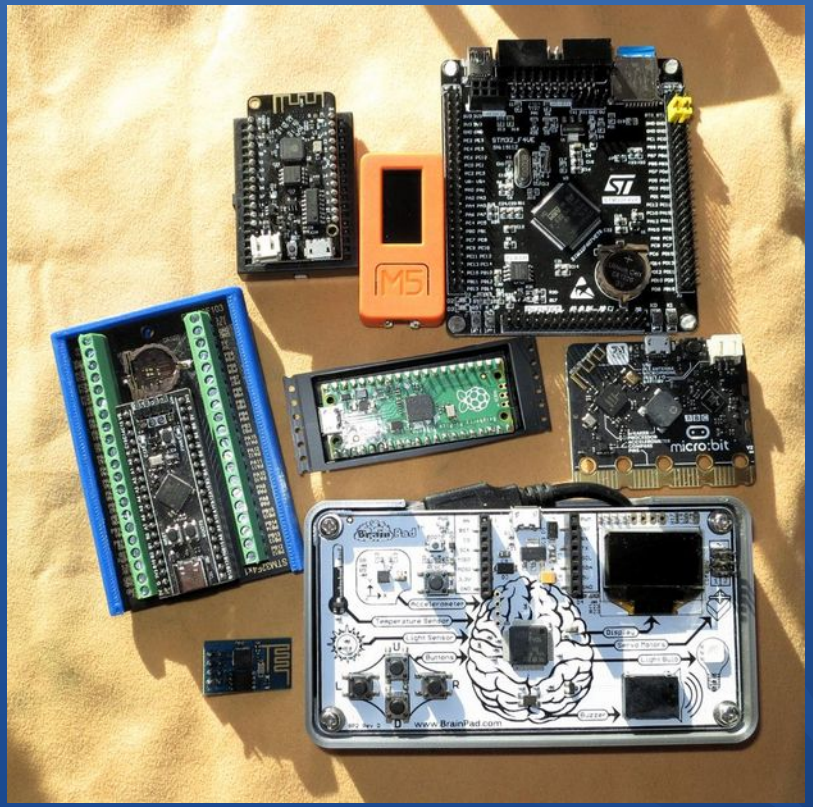
My Canada Post experience

- **BuyaPi**: ordered Jan 21, arrived Feb 2.
  Average speed: house spider
- **Elmwood**: ordered Jan 27, arrived Feb 3.
  Average speed: 3-toed sloth
- Don't use Canada Post because

What is MicroPython?

- Python 3 implementation
- Small: 256 KB flash, 16 KB RAM (minimum)
- Compiled on-chip; standalone
- Subset of standard Python library
- Core developers were hired to implement for Pico
- Now includes ARMv6M assembler

micropython.org



MicroPython Differences

- System libraries are typically limited, e.g.:
  - Strings are always UTF-8; 8-bit codecs excluded
  - Time is monotonic (fractional) seconds: no timezones or DST
  - No CSV, numpy, pip (→ upip), …
- .py → .mpy (like .pyc) compilation isn't automatic
- Hardware interface modules:
  - machine: for hardware features like pins, PWM, I^2C, ADC, …
  - rp2: RP2040 PIO assembler, raw Flash access
- help( ) docstrings short or absent: see online docs

Flashing MicroPython

- Pico firmware is distributed as UF2 images
- Hold BOOTSEL while plugging in
- Pico appears as a USB storage device
- Drag/copy UF2 to PICO storage
- Pico reboots; USB disappears

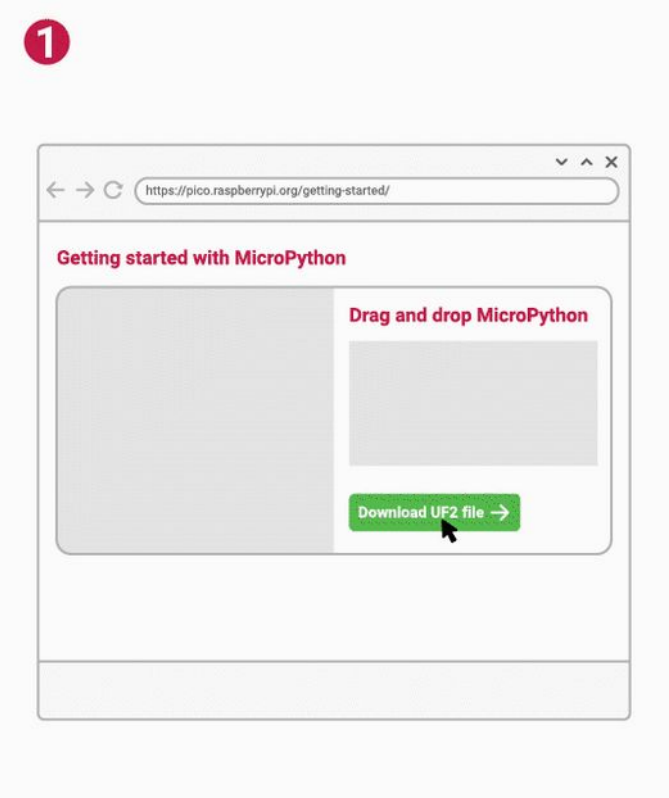
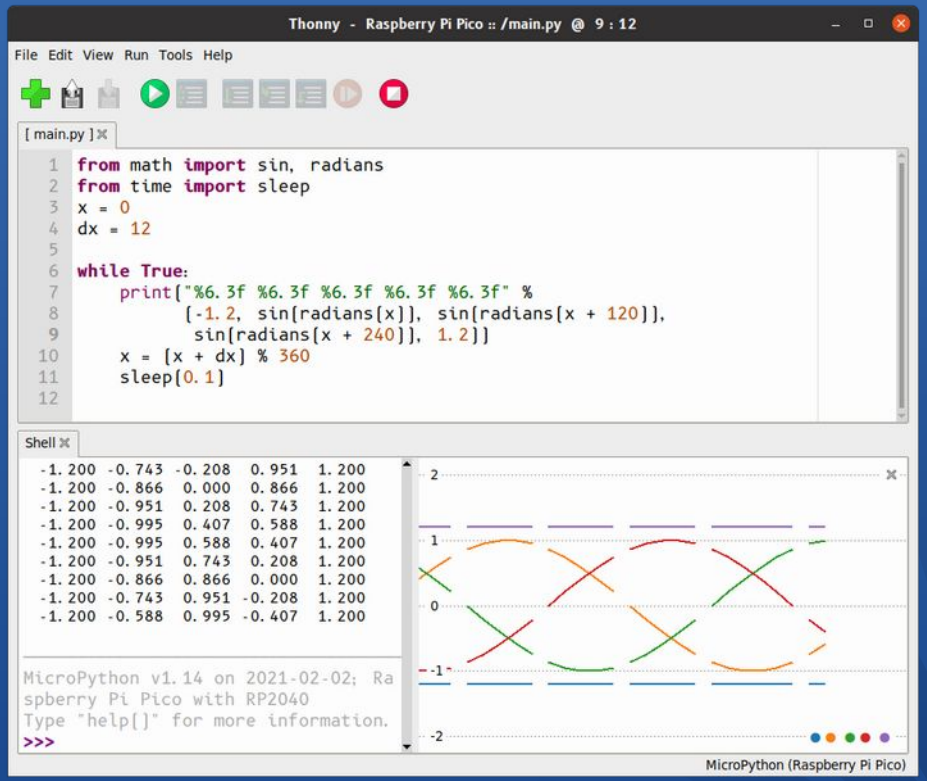


Image credit: Raspberry Pi Foundation

Editing: Thonny

- Raspberry Pi Foundation's recommended editor
- Installed by default
- Includes loading/saving to Pico flash
- Has a simple graph tool
- … plus firmware updater
- … and (Raspberry) REPL
  sorry not sorry

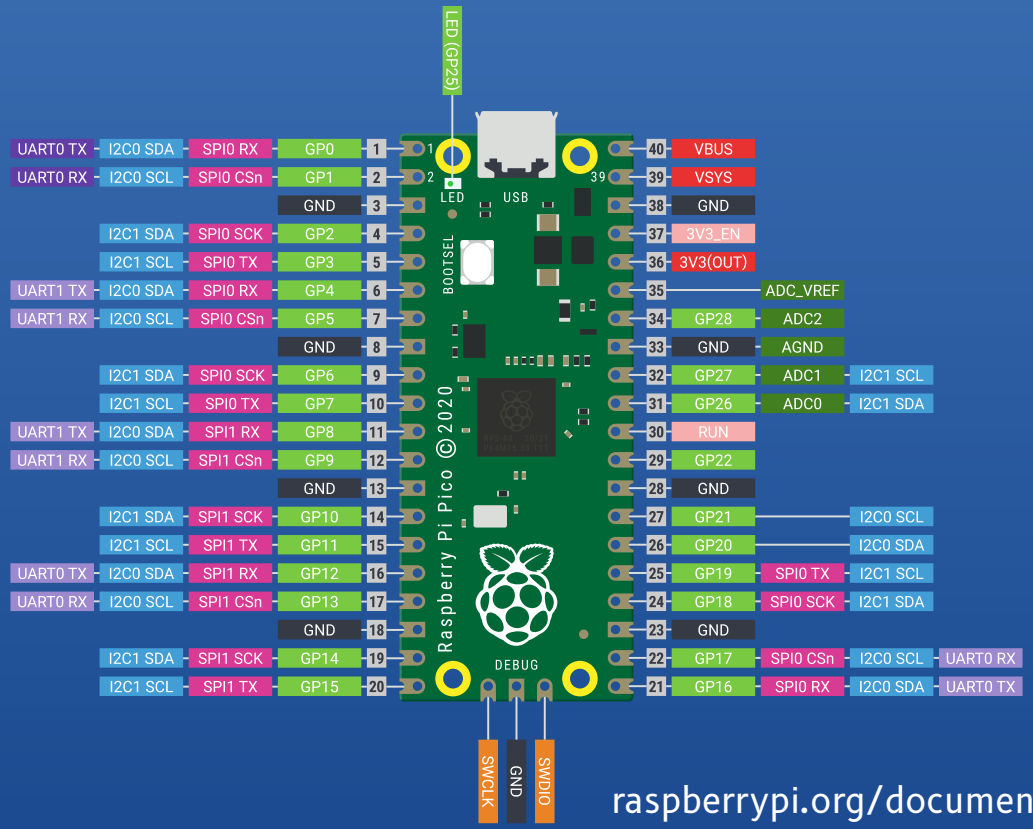


(and no, I don't know why the graph broke)

“but my $EDITOR …!!1!”

- You don’t have to use Thonny
- … it’s just more work if you don’t.
- The command-line MicroPython tool with REPL access is rshell: https://github.com/dhylands/rshell/tree/pico
- Make sure you get the this branch, as it handles the quirks of the Pico’s RTC
- … and yes, the Foundation has shipped yet another device which doesn’t have battery backup on its clock ☹

All of the Pins

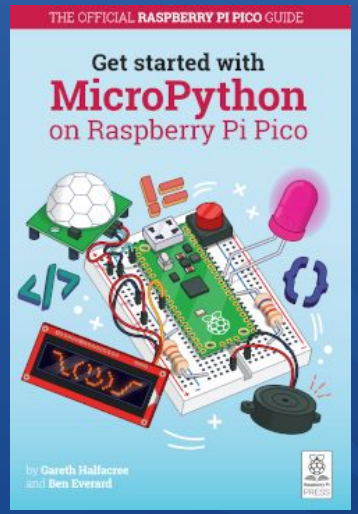


raspberrypi.org/documentation/pico/getting-started

Power | Ground | UART / UART (default) | GPIO, PIO, and PWM | ADC | SPI | I2C | System Control | Debugging

Documentation

- This is absolutely *stellar* for a board at launch + 3 weeks
- Data sheets, API guides, code, Fritzing parts … all at raspberrypi.org/documentation/pico/getting-started



Unexpected Quirks

It's a new board, and folks are just learning, but:

1. ADC: default analogue-to-digital setup is quite noisy
2. PWM: duty cycle changes if frequency is changed
3. UART Serial: has no wait/timeout, will lock if read and no data waiting
4. Dual core/threading: seems to be not well understood yet

Worked example

- Trevor Woerner used an MCP3008 with a thermistor and Raspberry Pi last month:
- Let's use a Pico
- … which is cheaper than an MCP3008 (by 50¢ !)
- … and can act as a serial datalogger, perhaps writing to a Raspberry Pi over USB serial.

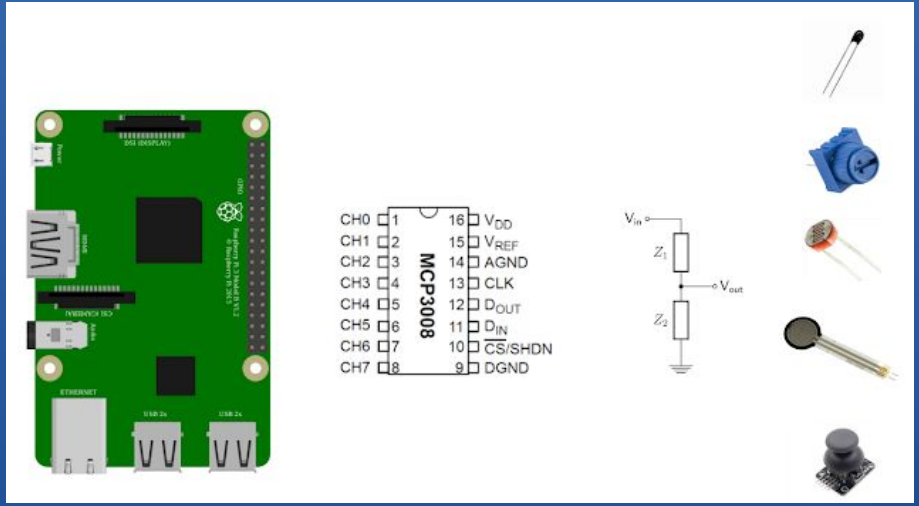
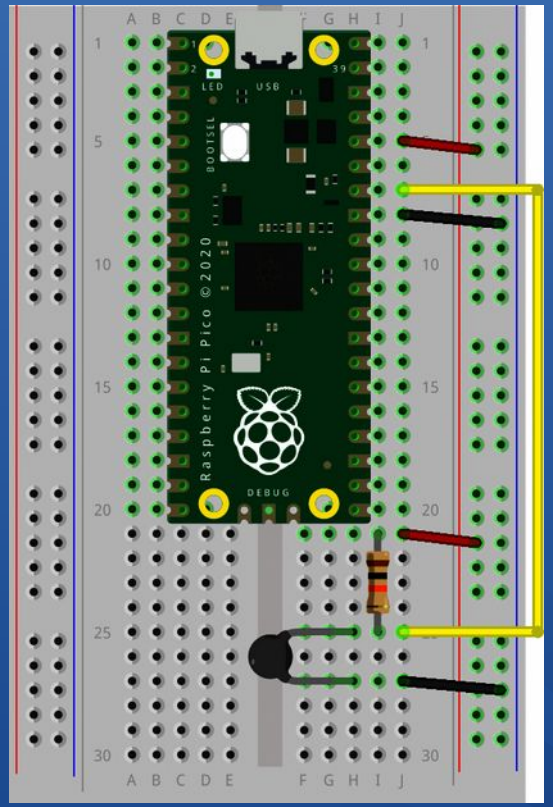


Image credit: Trevor Woerner
twoerner.blogspot.com/2021/01/sensing-temperature-with-raspberrypi.html

Wiring

- Pins used:
  - ADC2 (pin 34)
  - 3V3 (pin 36)
  - AGND (pin 33)
- 10 kΩ resistor between 3V3 and thermistor
- 10 kΩ @ 25 °C thermistor, β = 3977



Code!

```
from machine import Pin, ADC
from time import sleep
from math import log

led = Pin(25, Pin.OUT)
adc = ADC(2)
r25 = 10000
beta = 3977

while True:
    r = 10000.0 / (65535 / float(adc.read_u16()) - 1)
    lnr = log(r / r25)
    ts_C = -273.15 + 1/(1/298.15 + lnr/beta)
    print('%5.1f' % (ts_C))
    led.toggle()
    sleep(2)
```