

HackEDbeta 2024

Introduction to the

Raspberry Pi Pico

Steven Knudsen, PhD PEng
knud@ualberta.ca



DEPARTMENT OF

Electrical and Computer Engineering



Innovation
Creativity
Entrepreneurship
Incubator
www.iceincubator.com



Outline

1. Raspberry Pi Pico and Pico W overview
 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/HackEDbeta2024Workshop



Assumptions

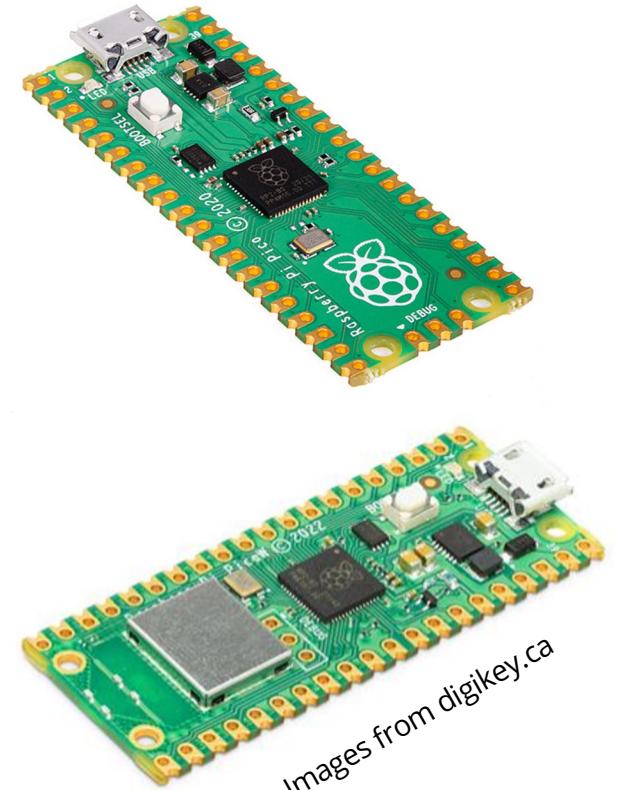
- You know how to program in at least one of the languages
 - Python or C
- You have a Pico
 - You have two if you want to program in Rust



Raspberry Pi Pico and Pico W

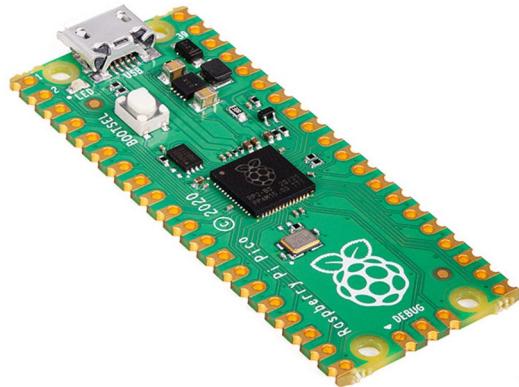
Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- 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



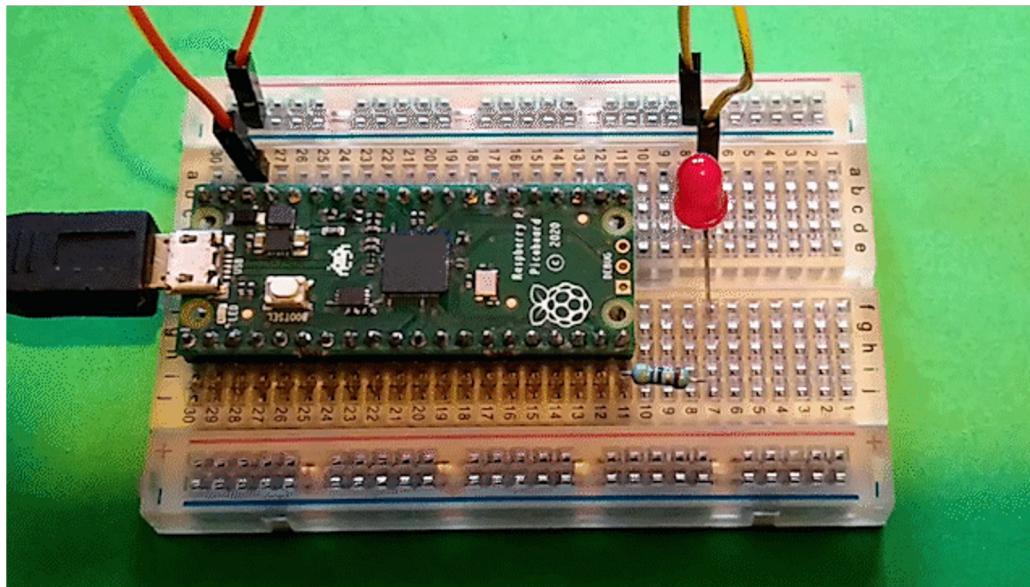
\$CA8.34 at canada.newark.com

Images from digikey.ca

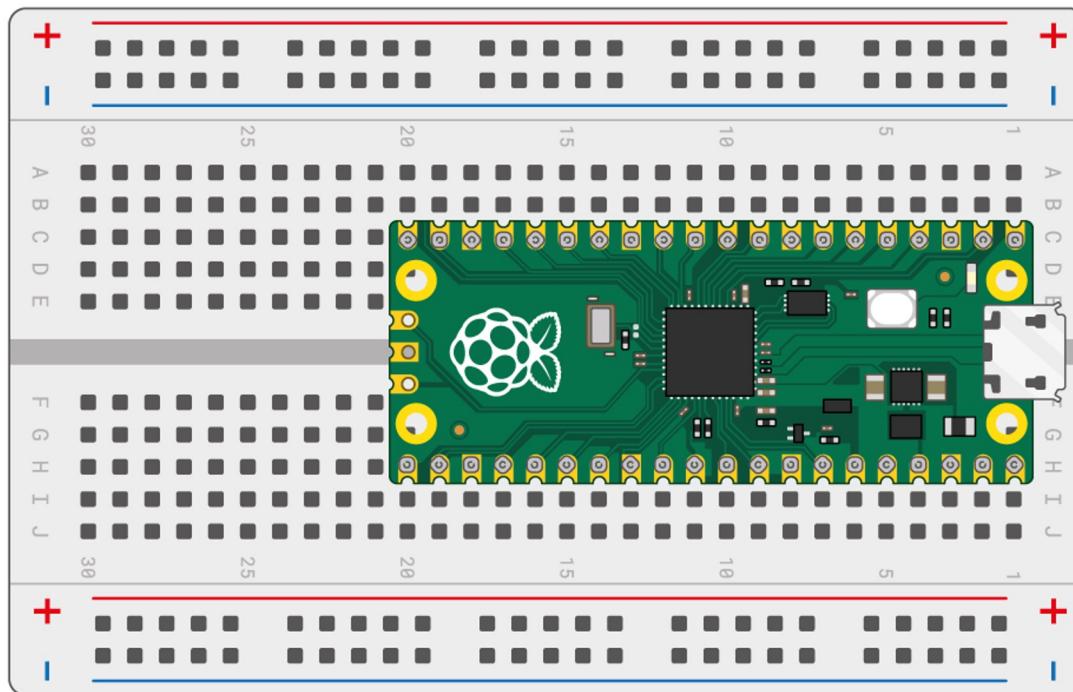
Getting started with the Pico - MicroPython

The main reference is a very good tutorial

<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/0>

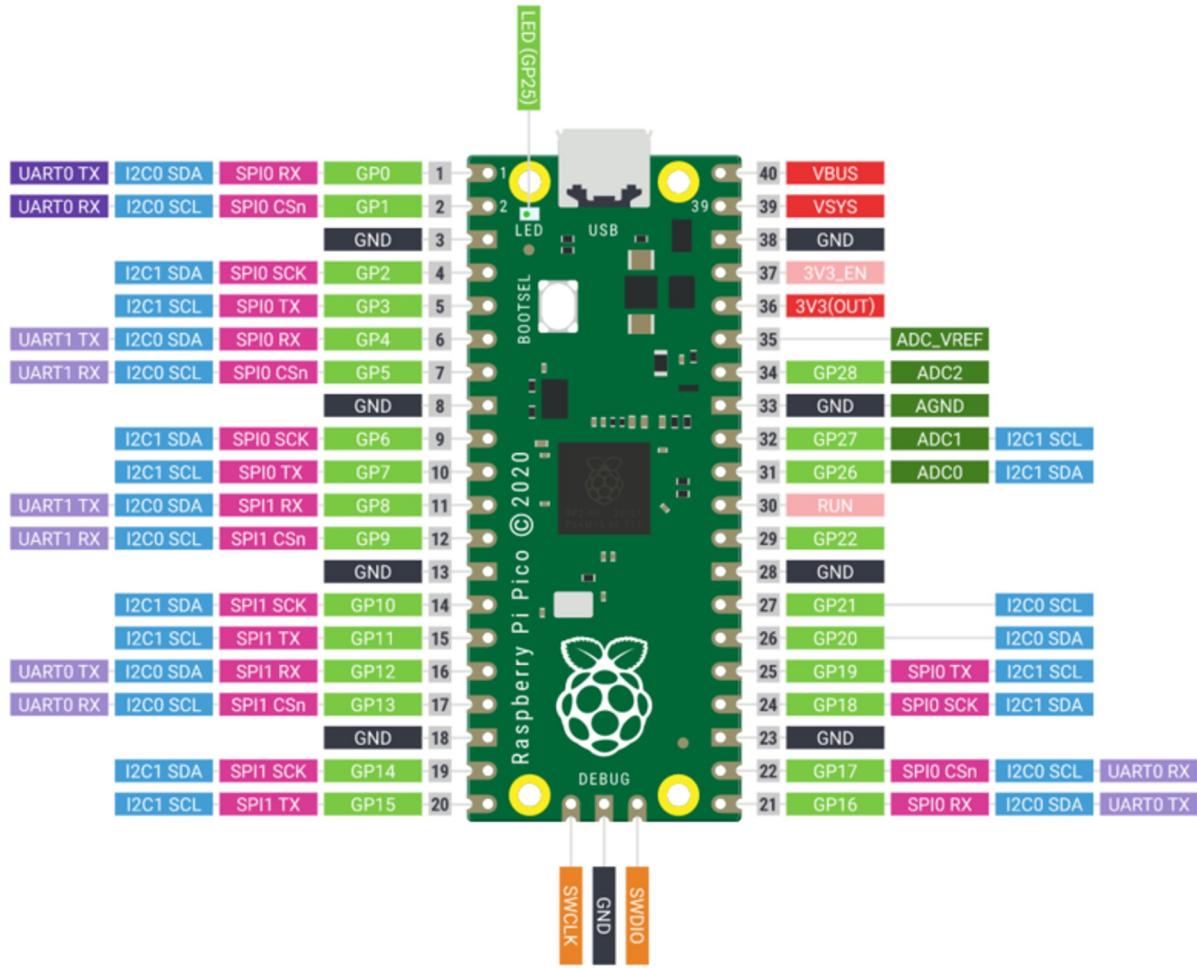


Pico on breadboard



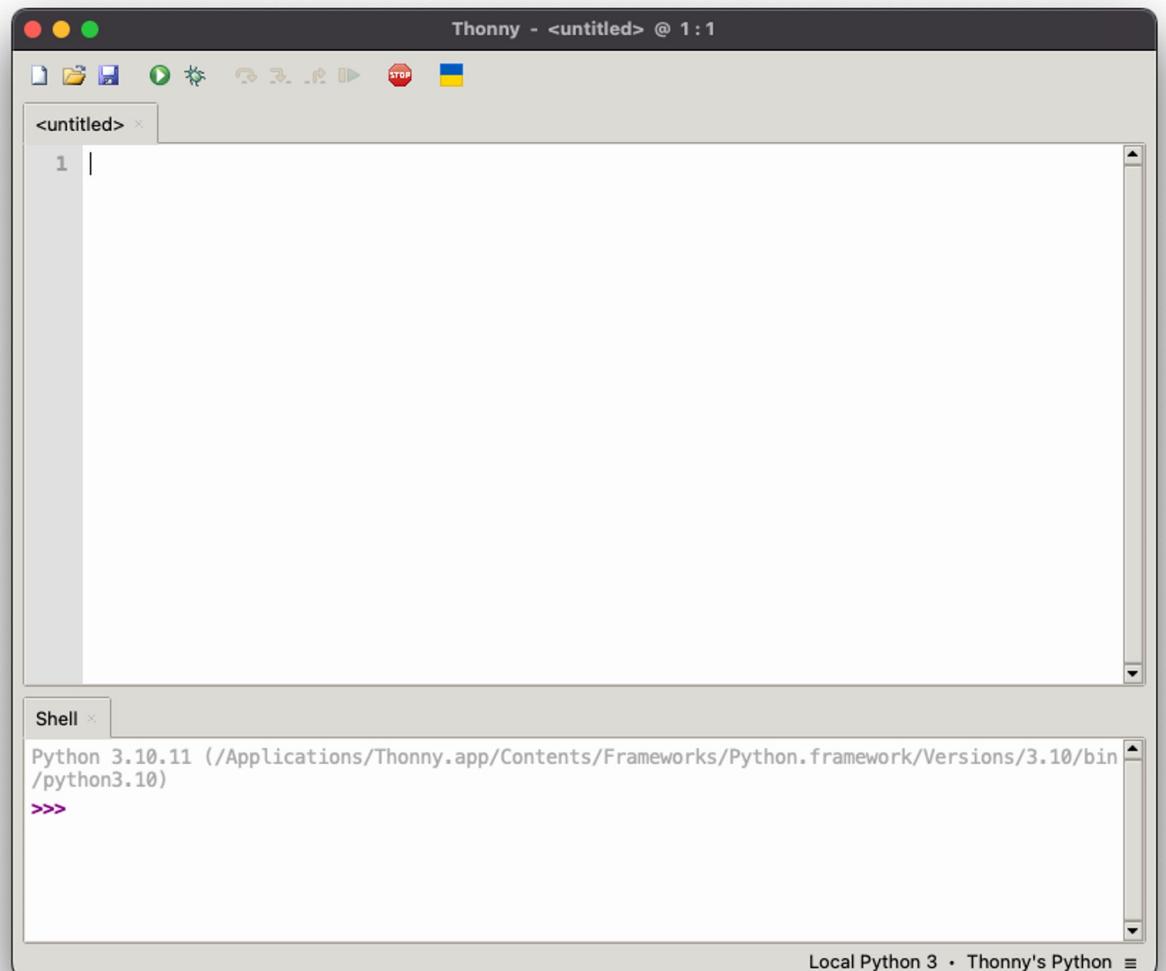
<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/1>

RPi Pico Pins



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

Install Thonny (Win/Lin/macOS)



Pico in boot select (bootsel) mode

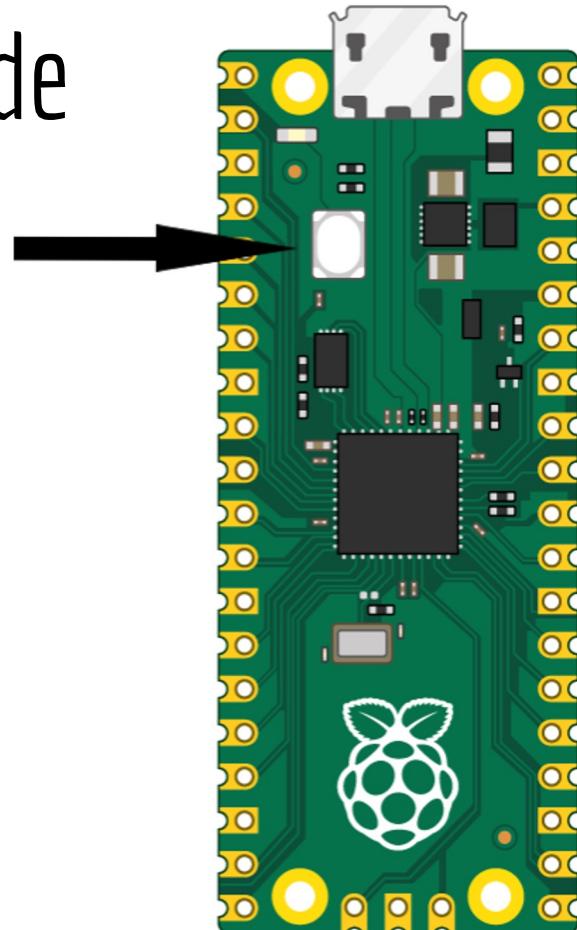
Unplug Pico from USB

Press and hold bootsel button

Plug Pico into USB and wait a few seconds

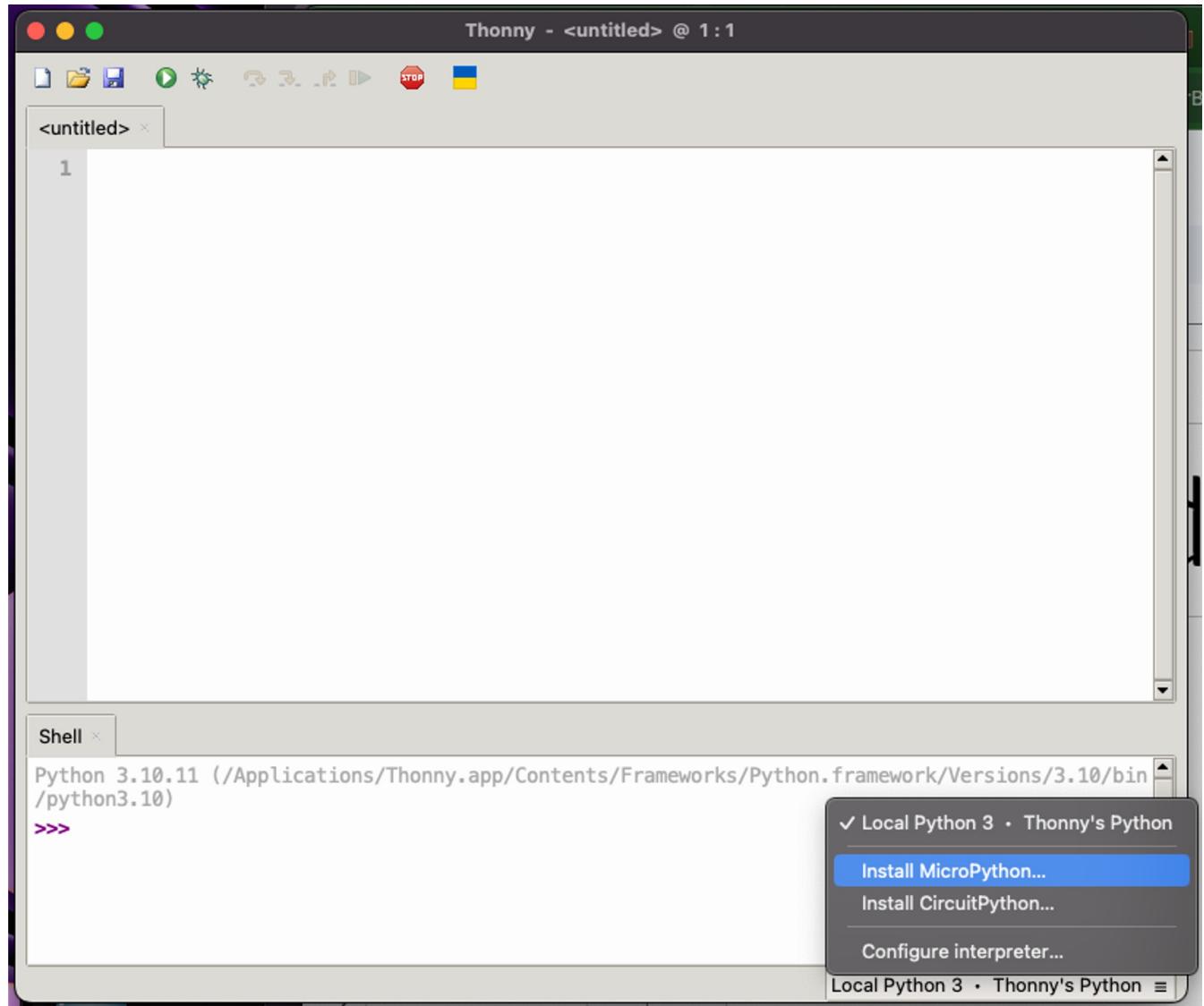
Release bootsel button

Open Thonny

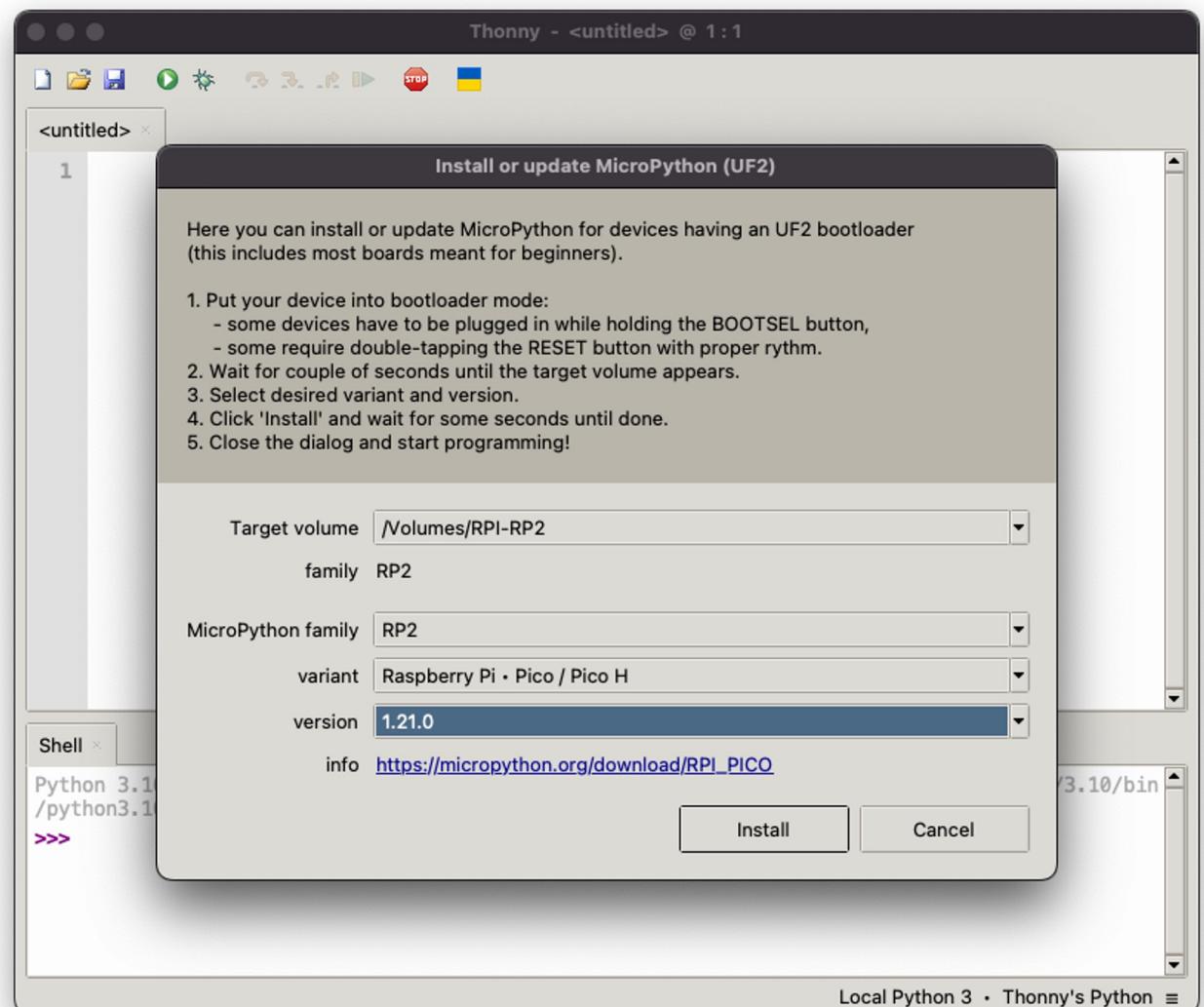


<https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3>

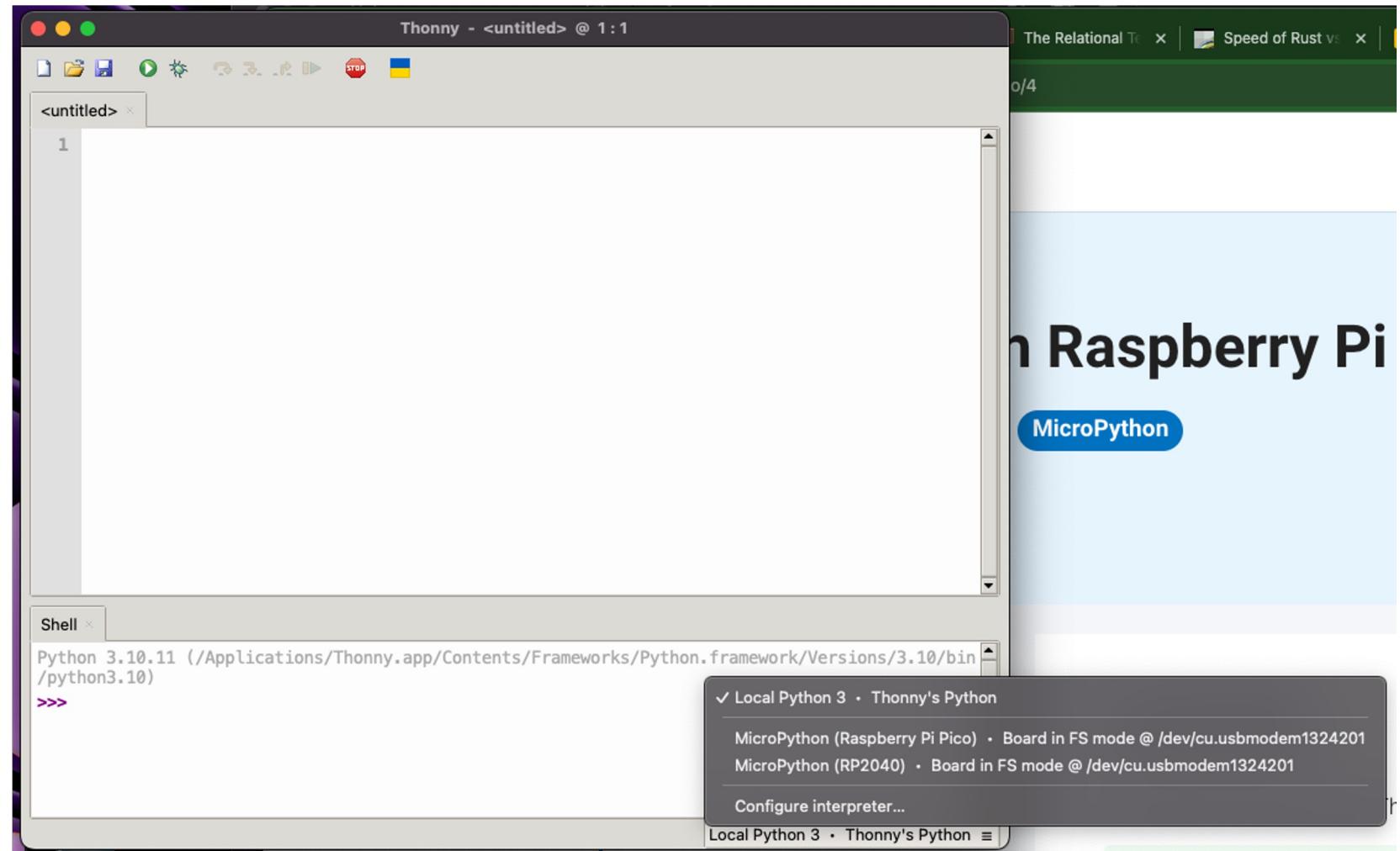
Install MicroPython

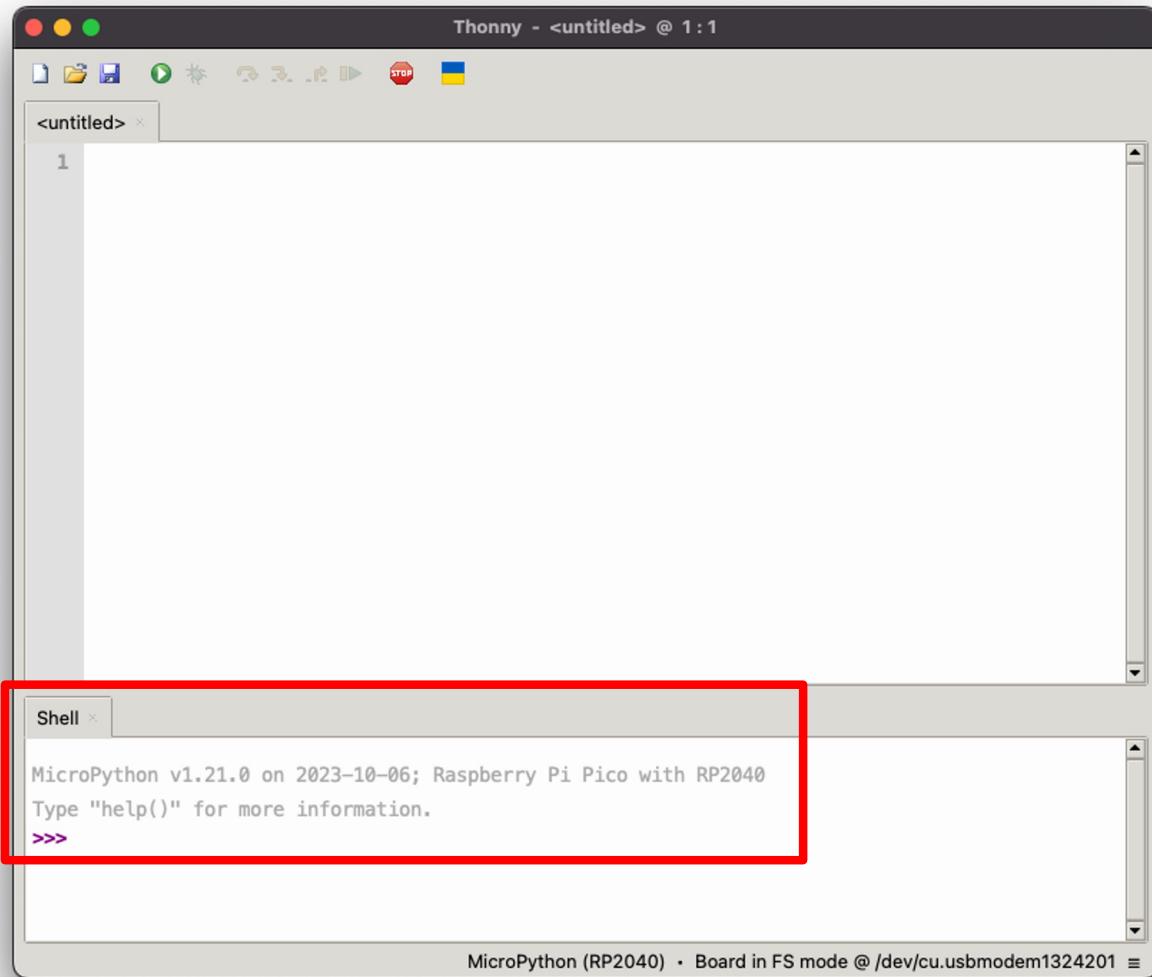


Install dialogue

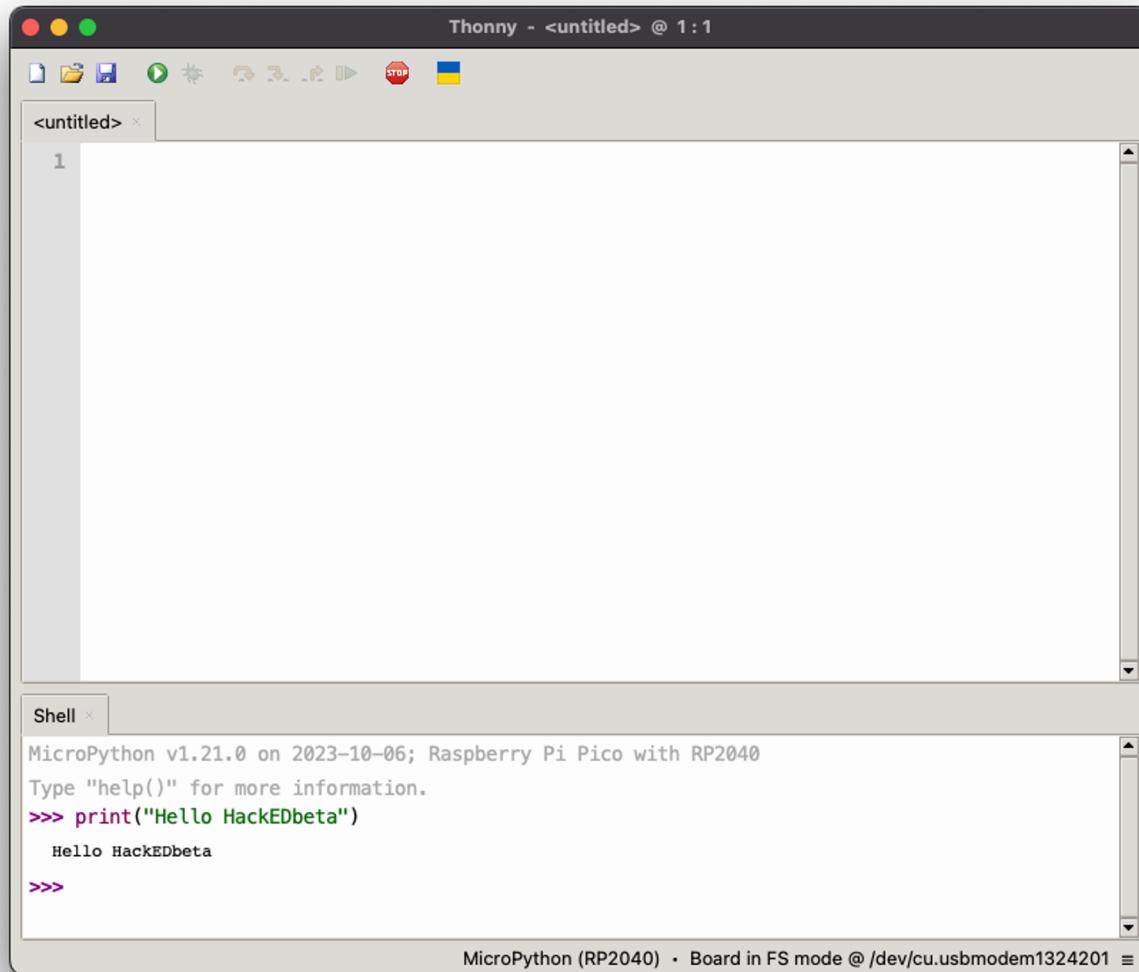


Connect





Success!



GPIOs

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

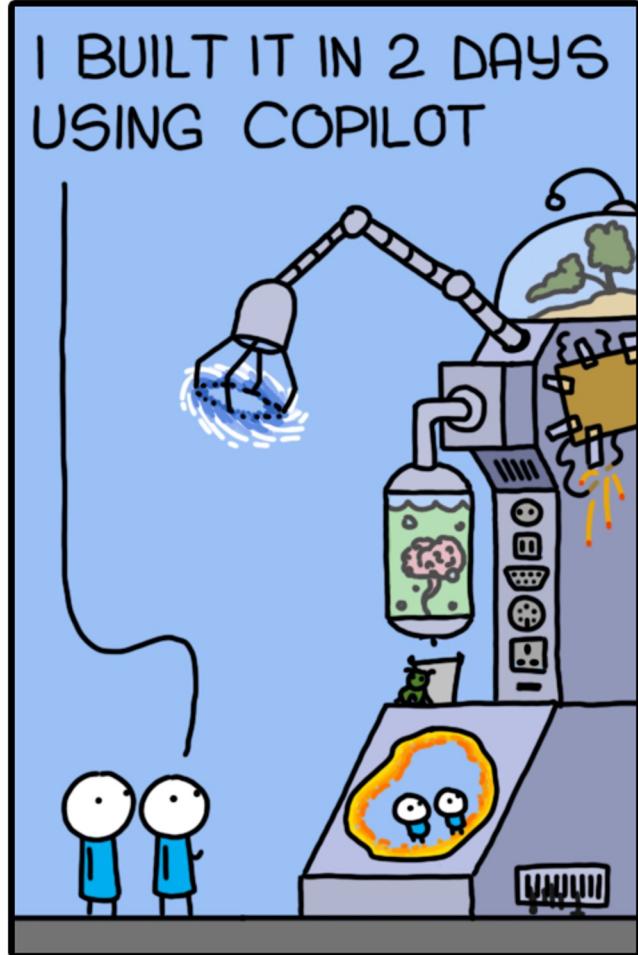
Quick Reference for MicroPython is found at

<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

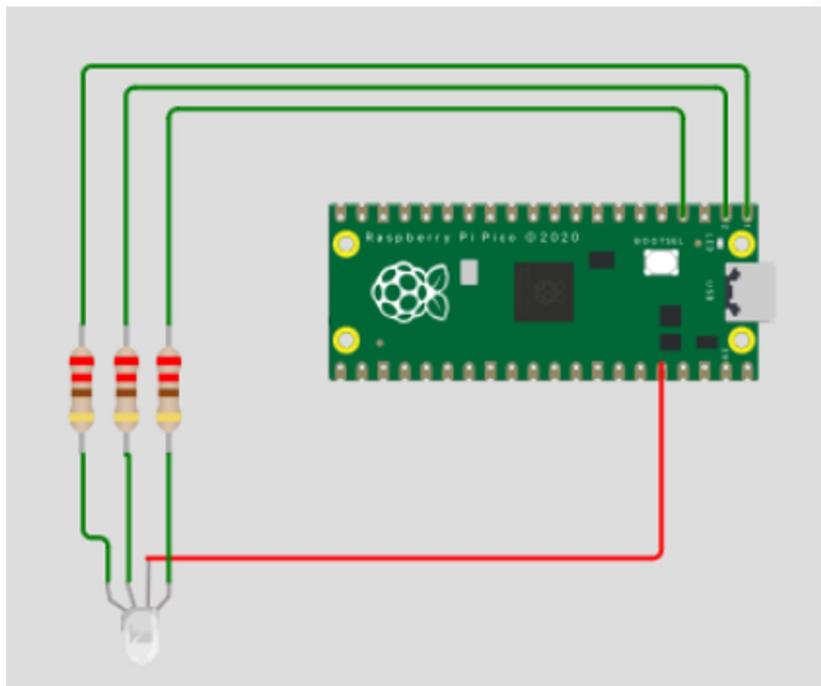
WHAT DOES IT DO?

I DON'T
KNOW...



monkeyuser.com

Simple RGB blinky



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

```
HackEDbetaPico01.py
```

```
1 import rp2
2 import time
3
4 from machine import Pin
5
6 p0 = Pin(0, Pin.OUT)
7 p1 = Pin(1, Pin.OUT)
8 p2 = Pin(2, Pin.OUT)
9
10 while True :
11     p0.off() # pull low to turn on LED
12     p1.on() # pull high to turn off LED
13     p2.on() # pull high to turn off LED
14     time.sleep(0.25)
15     p0.on() # pull high to turn off LED
16     p1.off() # pull low to turn on LED
17     p2.on() # pull high to turn off LED
18     time.sleep(0.25)
19     p0.on() # pull high to turn off LED
20     p1.on() # pull high to turn off LED
21     p2.off() # pull low to turn on LED
22     time.sleep(0.25)
```

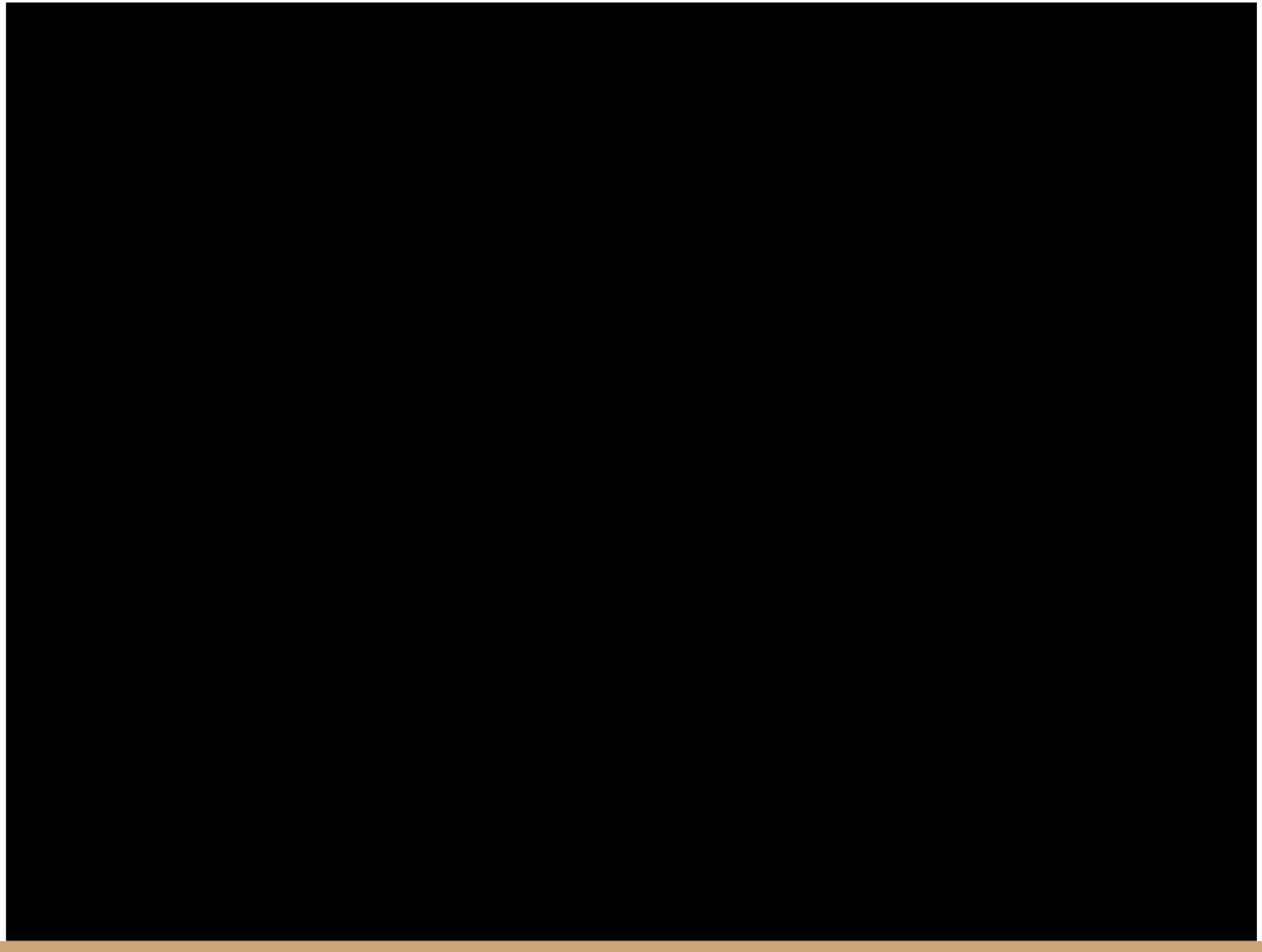
Shell

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

```
MPY: soft reboot
```

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

Live demo



Getting started with the Pico - 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 Pico examples

```
git clone https://github.com/raspberrypi/pico-examples.git
```

Follow the Getting Started

Follow the Getting Started Guide...

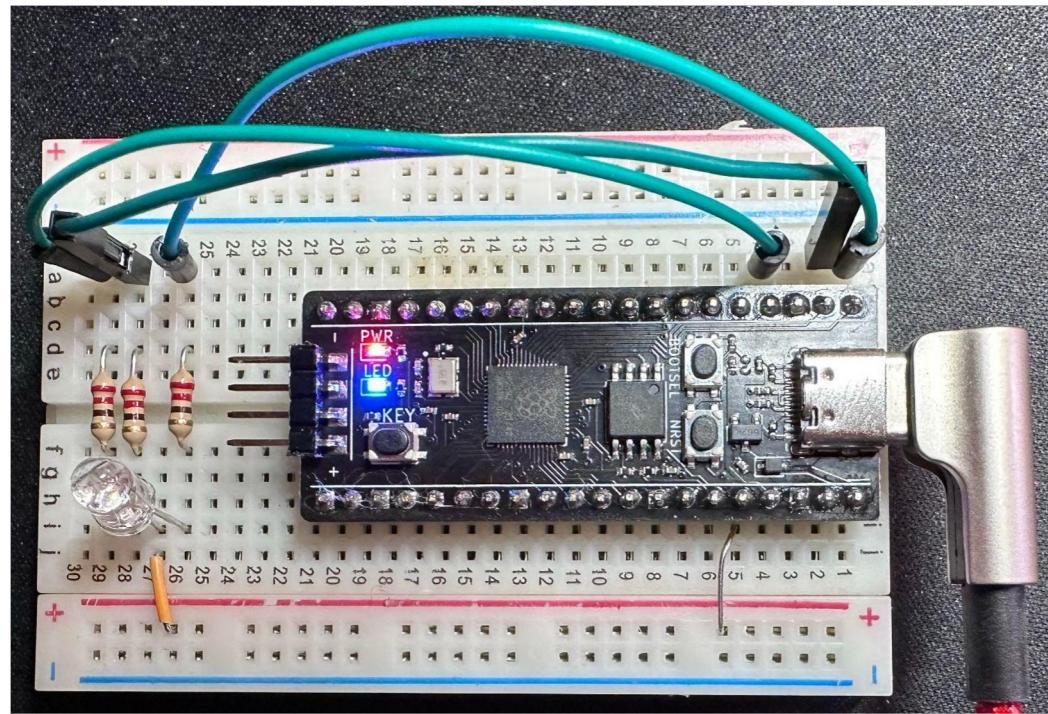
Do the “blink” example to be sure
the tool chain is up and running.

For this example I use an RP2040 board from Universal Solder.

Pressing BOOTSEL while plugging it in results in it looking like a storage device. Programming is drag'n'drop

Different Pico

From Universal Solder



Blink example

Let's make sure the blink example works

```
cd ~Development/RPi/pico-examples  
mkdir build; cd build  
cmake ..  
make -j4
```

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

Blink example

When complete, connect power to the Pico while holding the boot select button. The Pico will mount as a storage device. Drag and drop

`~Development/RPi/pico-examples/build/blink/blink.uf2` onto the Pico

The blue LED should blink

Replicate previous MicroPython Example

The main steps to make a custom C/C++ application are

1. In pico-examples copy blink folder and contents

```
cd ~/Development/RPi/pico-examples  
cp -dpR blink rgb_demo  
cd rgb_demo
```

2. Rename `blink.c` to `rgb_demo.c` and edit the `CMakeLists.txt` file replacing `blink` with `rgb_demo`

Replicate previous MicroPython Example cont'd

3. In the pico-examples directory, edit the `CMakeLists.txt` file
 - a. Find where the `blink` example is added
 - b. Replicate those lines and replace `blink` with `rgb_demo`
4. Change directory in to the build directory and execute `cmake ..`
5. There should be a new directory named `rgb_demo`. Change directory into it and execute

```
make -j4
```

3. There should be a new executable named `rgb_demo`.
4. Put the Pico back into bootloader mode and drop the `rgb_demo.uf2` file on the drive – the old blink demo should work

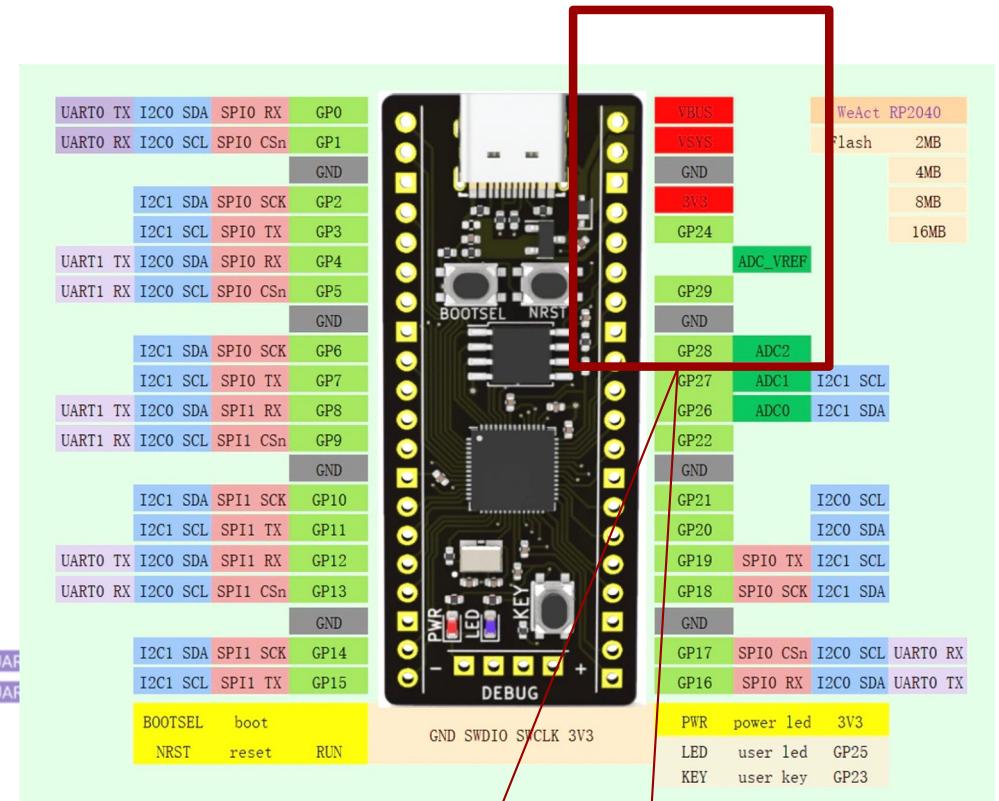
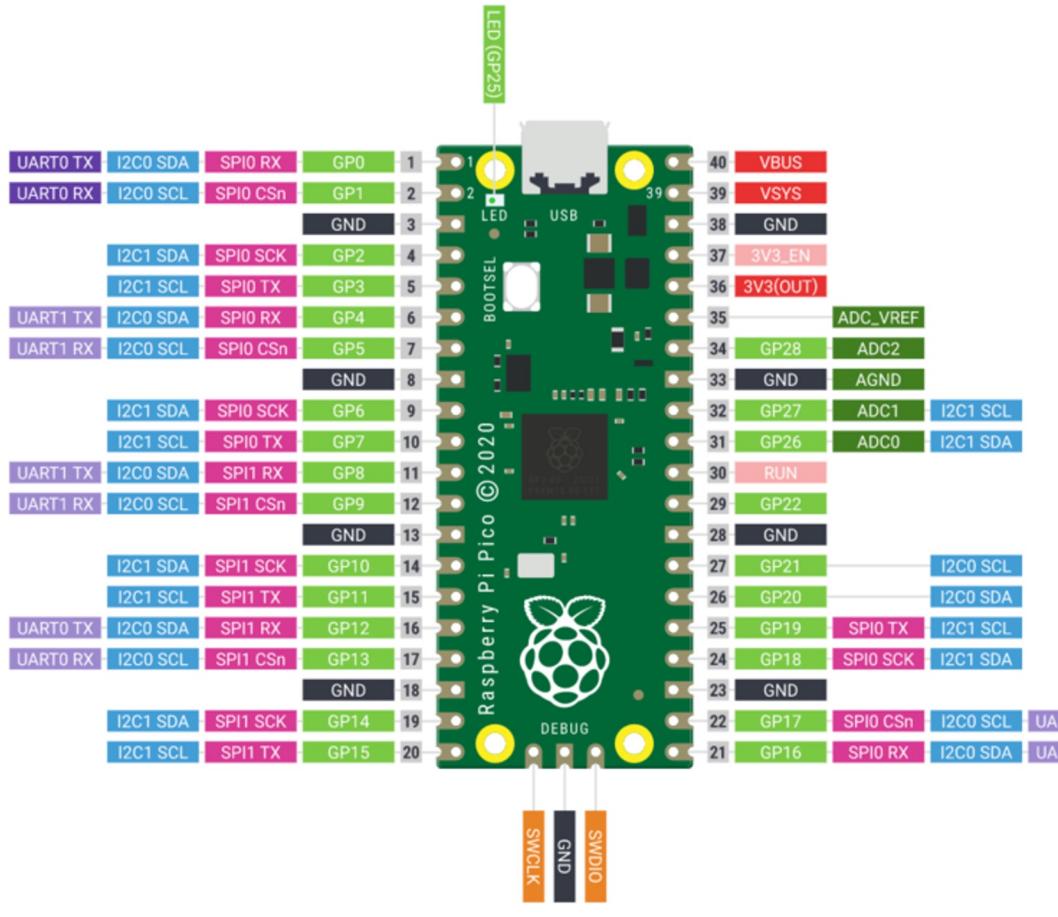
Let's replace the `rgb_demo.c` source with the equivalent of the python

```
#include "pico/stdlib.h"
#include "hardware/gpio.h"

const uint8_t RGB_PIN1 = 0;
const uint8_t RGB_PIN2 = 1;
const uint8_t RGB_PIN3 = 2;

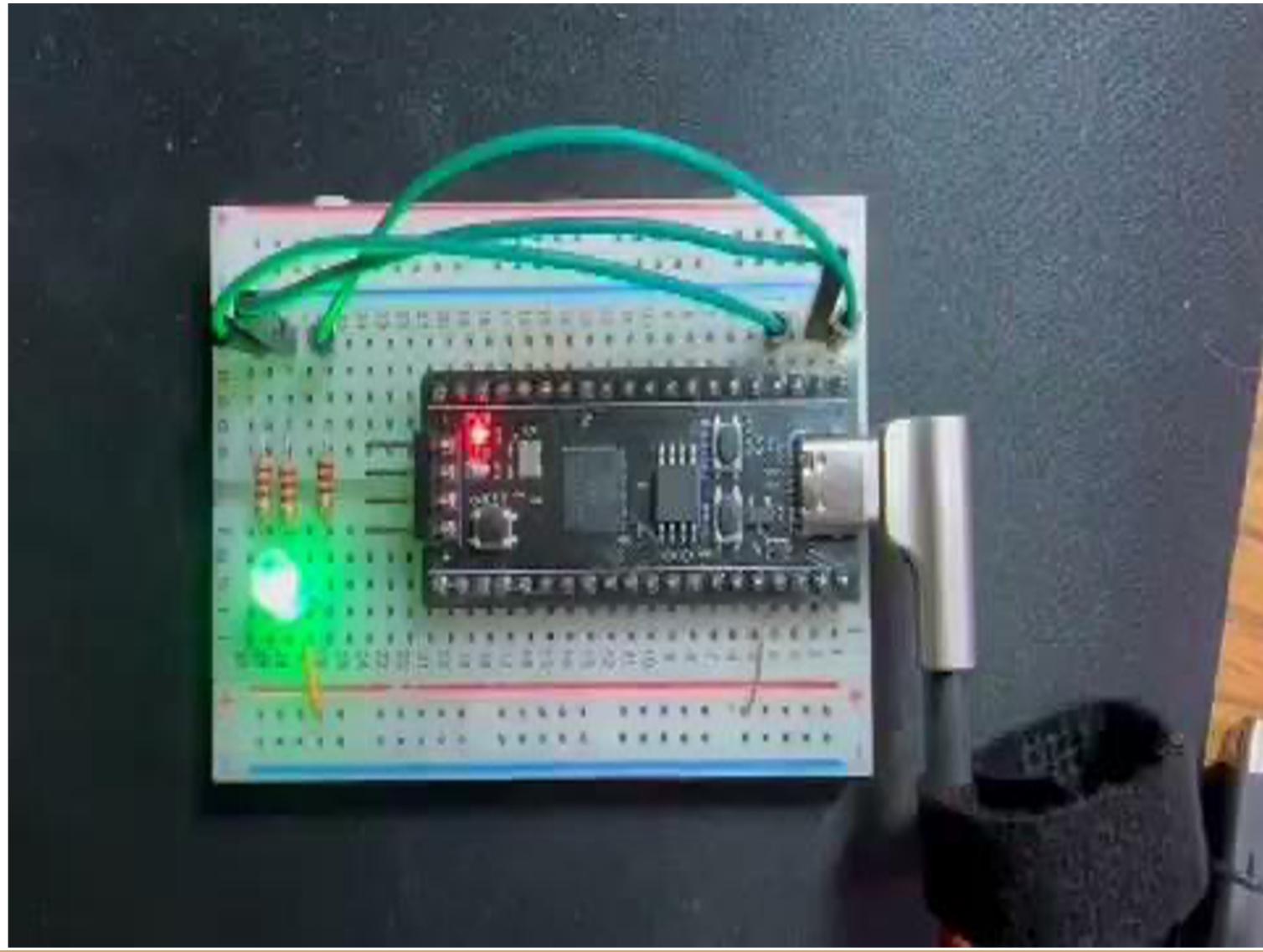
int main() {
    gpio_init(RGB_PIN1);
    gpio_set_dir(RGB_PIN1, GPIO_OUT);
    gpio_init(RGB_PIN2);
    gpio_set_dir(RGB_PIN2, GPIO_OUT);
    gpio_init(RGB_PIN3);
    gpio_set_dir(RGB_PIN3, GPIO_OUT);
    while (true) {
        gpio_put(RGB_PIN1, 0);
        gpio_put(RGB_PIN2, 1);
        gpio_put(RGB_PIN3, 1);
        sleep_ms(250);
        gpio_put(RGB_PIN1, 1);
        gpio_put(RGB_PIN2, 0);
        gpio_put(RGB_PIN3, 1);
        sleep_ms(250);
        gpio_put(RGB_PIN1, 1);
        gpio_put(RGB_PIN2, 1);
        gpio_put(RGB_PIN3, 0);
        sleep_ms(250);
    }
}
```

GOTCHA!



NOT THE SAME!

Live demo



What about Rust

Doable

However, we don't have time today to go through all the steps, it's involved!

<https://reltech.substack.com/p/getting-started-with-rust-on-a-raspberry>

Happy to help you if you need it for HackEDbeta

One more tool - Wokwi.com





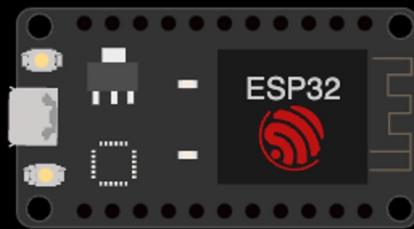
Simulate IoT Projects in Your Browser

[Discord Community](#) [LinkedIn Group](#)

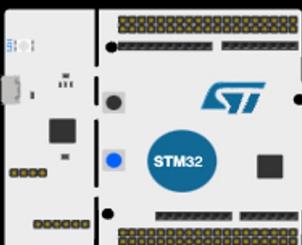
Simulate with Wokwi Online



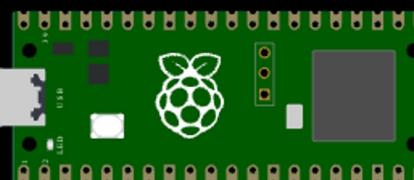
Arduino (Uno, Mega, Nano)



ESP32



STM32



Pi Pico

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

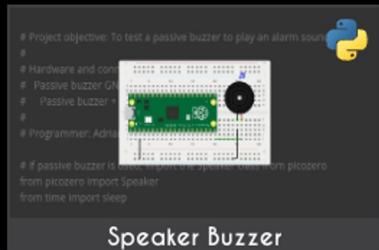
Featured projects



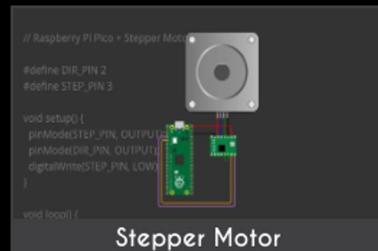
Traffic Light



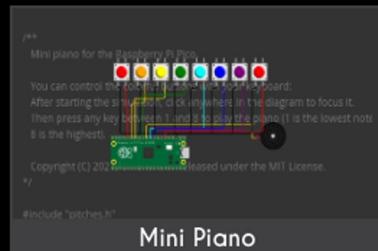
LCD 1602



Speaker Buzzer



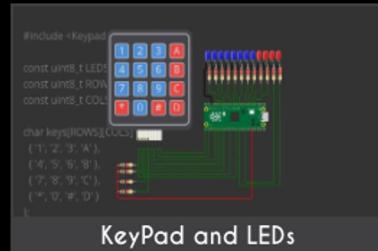
Stepper Motor



Mini Piano

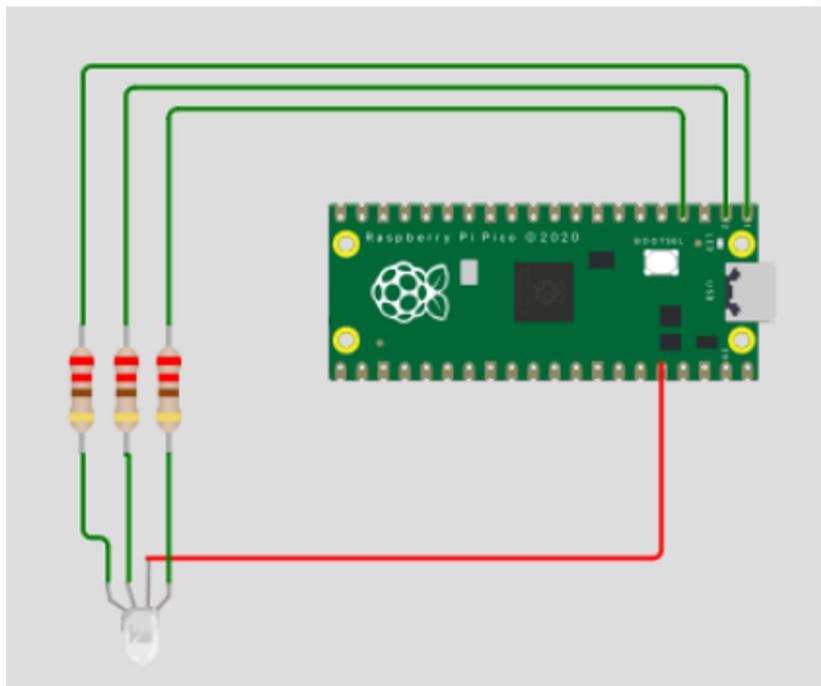


7 Segment Counter



Keypad and LEDs

Simple RGB blinky



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

```
HackEDbetaPico01.py
```

```
1 import rp2
2 import time
3
4 from machine import Pin
5
6 p0 = Pin(0, Pin.OUT)
7 p1 = Pin(1, Pin.OUT)
8 p2 = Pin(2, Pin.OUT)
9
10 while True :
11     p0.off() # pull low to turn on LED
12     p1.on() # pull high to turn off LED
13     p2.on() # pull high to turn off LED
14     time.sleep(0.25)
15     p0.on() # pull high to turn off LED
16     p1.off() # pull low to turn on LED
17     p2.on() # pull high to turn off LED
18     time.sleep(0.25)
19     p0.on() # pull high to turn off LED
20     p1.on() # pull high to turn off LED
21     p2.off() # pull low to turn on LED
22     time.sleep(0.25)
```

Shell

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

```
MPY: soft reboot
```

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

Wokwi Version

The screenshot shows the Wokwi web-based development environment. On the left, the code editor displays `main.py` with the following Python script:

```
1 import rp2
2 import time
3
4 from machine import Pin
5
6 p0 = Pin(0, Pin.OUT)
7 p1 = Pin(1, Pin.OUT)
8 p2 = Pin(2, Pin.OUT)
9
10 while True :
11     p0.off() # pull low to turn on LED
12     p1.on() # pull high to turn off LED
13     p2.on() # pull high to turn off LED
14     time.sleep(0.25)
15     p0.on() # pull high to turn off LED
16     p1.off() # pull low to turn on LED
17     p2.on() # pull high to turn off LED
18     time.sleep(0.25)
19     p0.on() # pull high to turn off LED
20     p1.on() # pull high to turn off LED
21     p2.off() # pull low to turn on LED
22     time.sleep(0.25)
23
```

On the right, the simulation window shows a Raspberry Pi Pico board with three LEDs connected to pins 0, 1, and 2. The LEDs are represented by small yellow circles with red horizontal stripes. The connections are shown as green lines: pin 0 connects to the first LED via a green line, pin 1 connects to the second LED via a green line, and pin 2 connects to the third LED via a green line. A red line also connects the ground rail to the common cathode of the three LEDs.

Trivia

...with wee prizes...



Trivia

Who created the first artificial neural network
and when?



- Warren McCulloch and Walter Pitts
- 1943
- Perceptron
- Modelled using electrical circuits

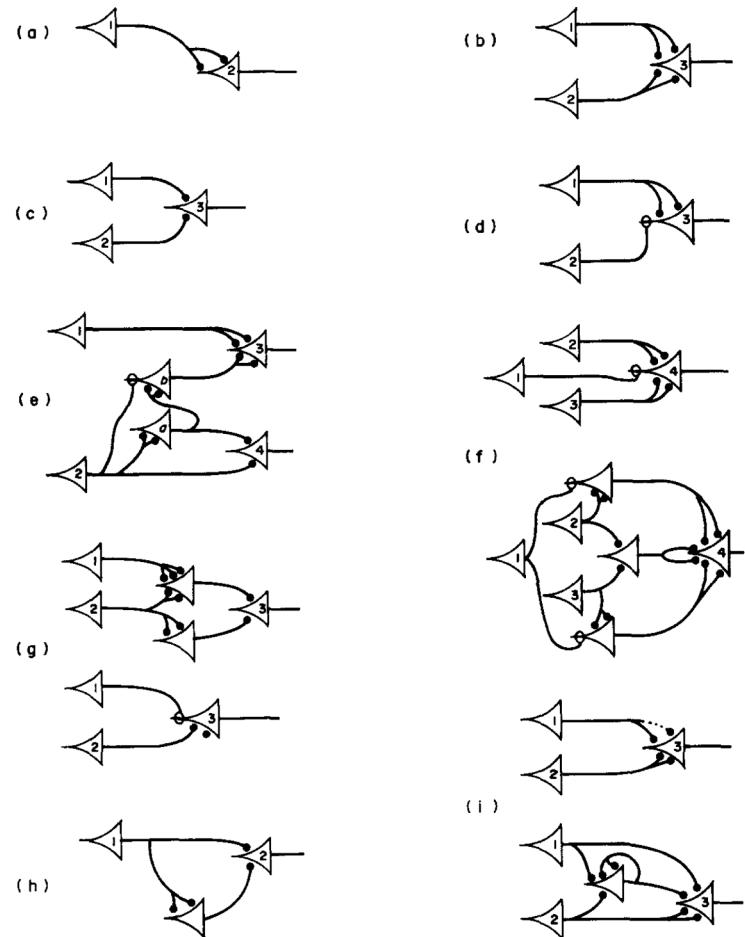


Figure 1. The neuron c_i is always marked with the numeral i upon the body of the cell, and the corresponding action is denoted by “ N ” with i subscript, as in the text:

Trivia

What is an equivalent to the U of A's main frame computer from the 80s?

Amdahl 470 v/6

Date Introduced : 1975

Dimensions overall: 63" x 70" x 26"

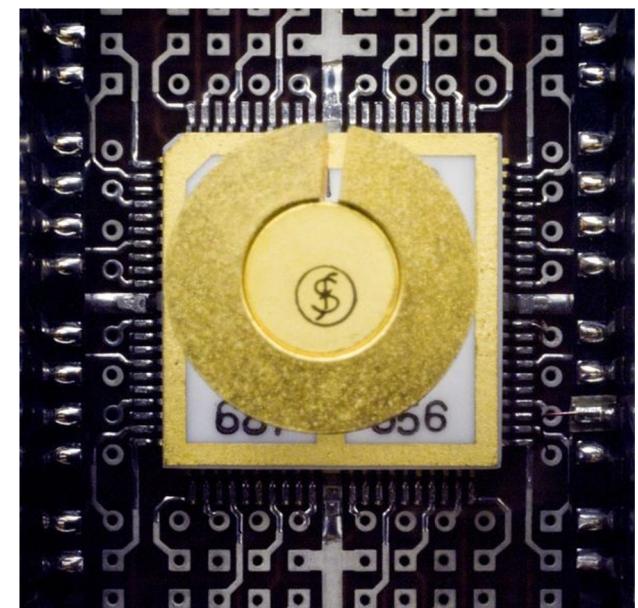
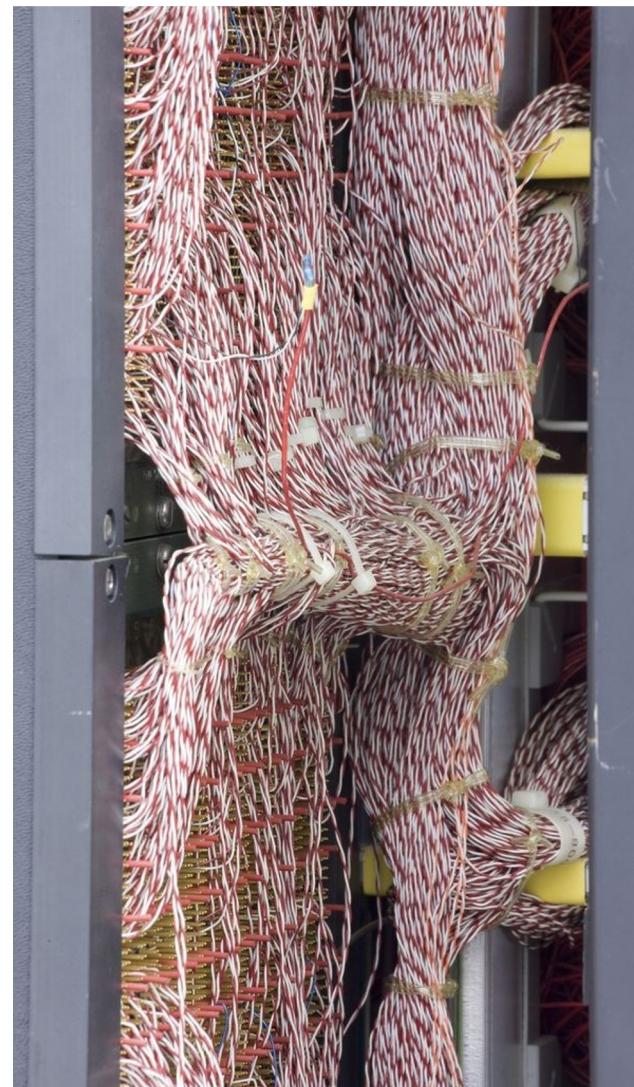
Keywords : Clones; Plug compat; IBM

Speed : 3.5 MIPS

Memory Size : up to 8MB

Memory Width : 32-bit

Cost : \$3,750,000 (2020 \$17,981,250)



Arduino

Date Introduced : 2010

Dimensions overall: 2.7" x 2.1" x 0.6"

Speed : 8 - 11 MIPS

Memory Size : 32k FLASH 2k SRAM

Cost : ~\$20

Cost to make : < \$5



Trivia

What are the most loved and hated languages
in the 2022 Dice survey

Most loved

Rust 86.83%

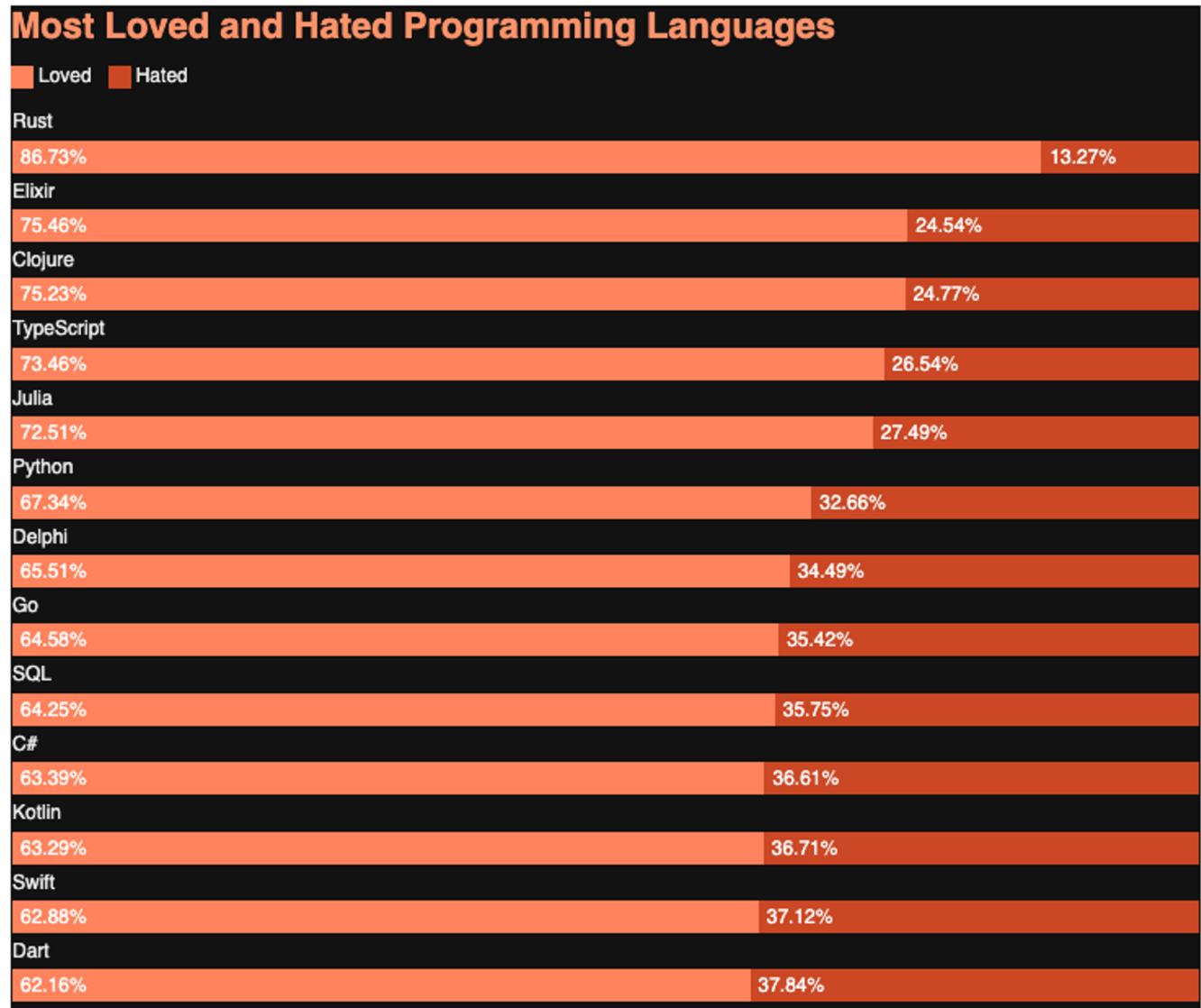
Elixir 75.46%

Clojure 75.23%

TypeScript 73.46%

Julia 72.51%

Python 67.34%

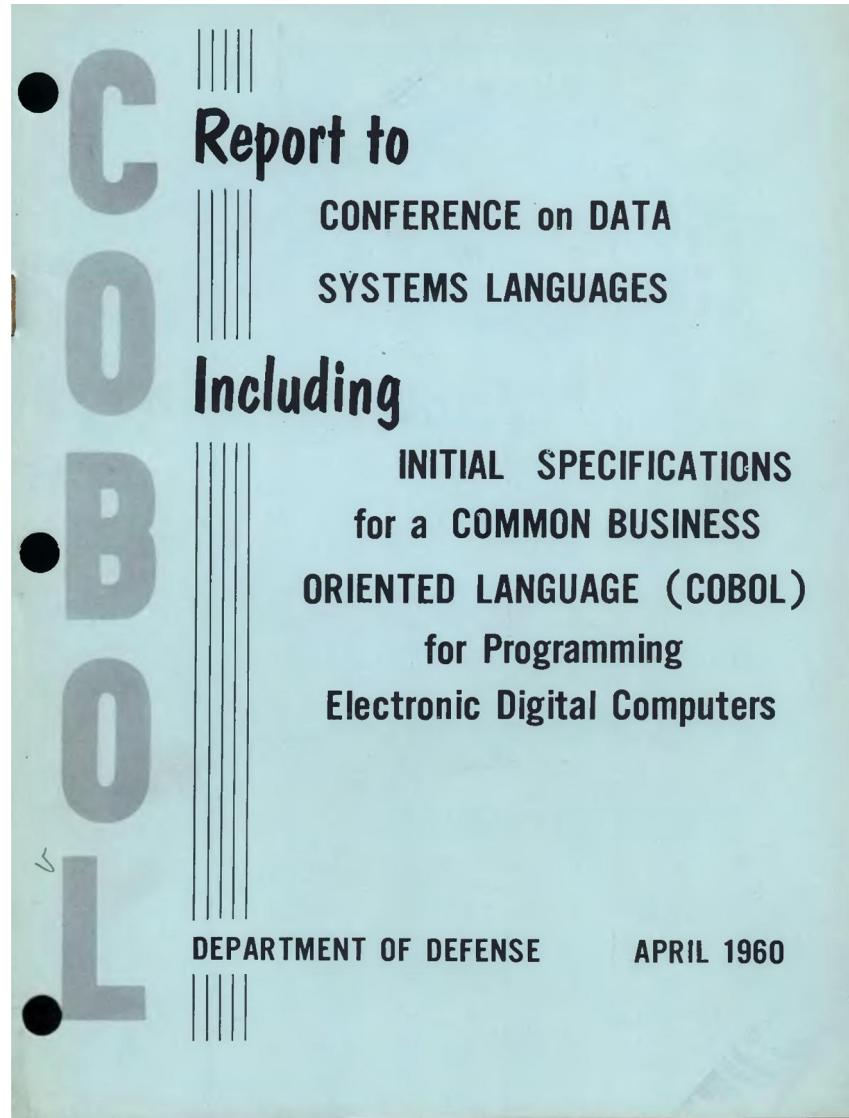


Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and

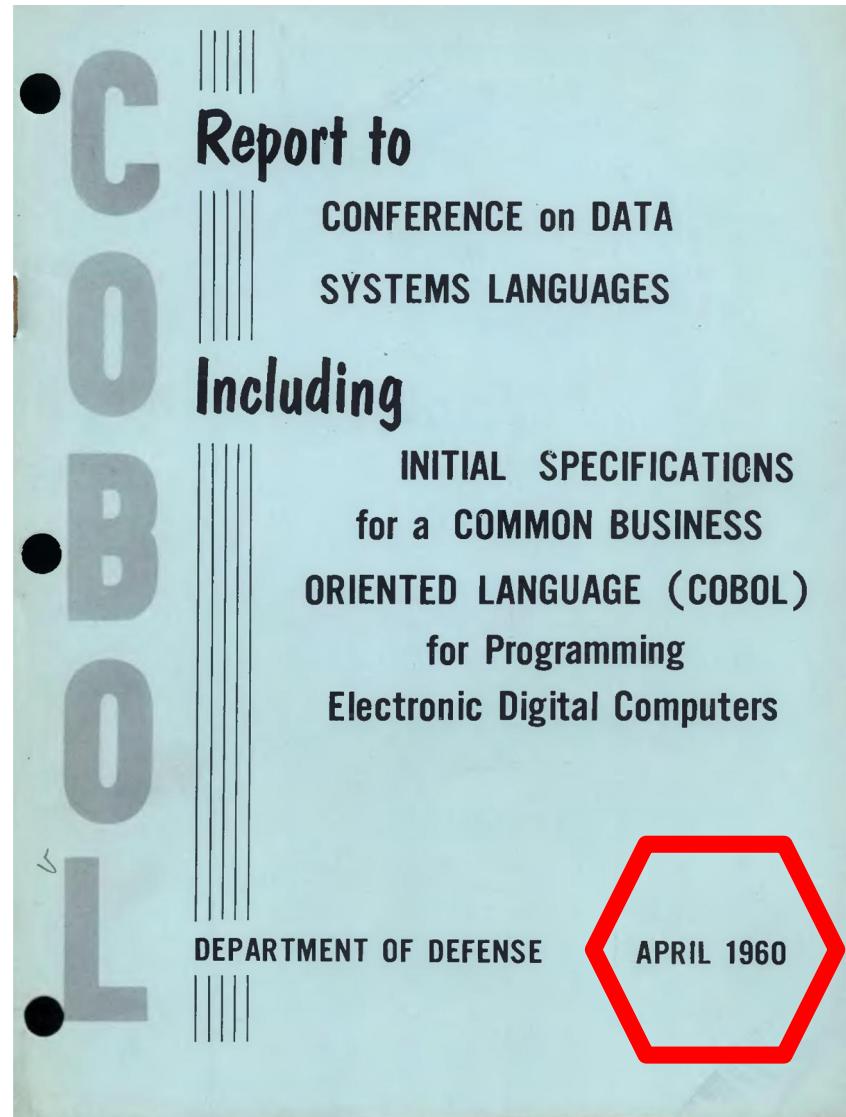


Most hated

VBA (78.56 percent).

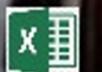
COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and



STACK OVERFLOW

**YOU ARE BY FAR THE WORST
PROGRAMMING LANGUAGE I HAVE EVER HEARD OF
VISUAL BASIC**



BUT YOU HAVE HEARD OF ME