# HackEDbeta 2023
# Introduction to the Raspberry Pi Pico

Steven Knudsen, PhD PEng
knud@ualberta.ca

ENGINEERING ALBERTA

DEPARTMENT OF

Electrical and C Engineering

ENGINEERING ALBERTA

**I**nnovation
**C**reativity
**E**ntrepreneurship
**Incubator**
*www.iceincubator.com*

ISAIC

# 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/knud/HackEDbeta2023Workshop

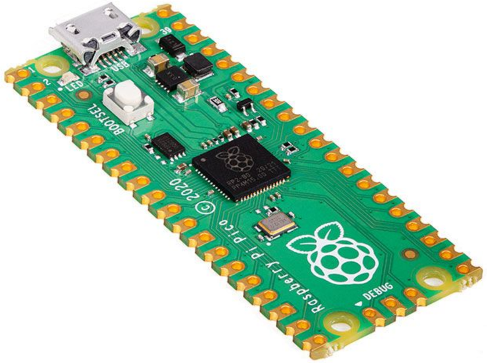
# Assumptions

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

# Raspberry Pi Pico and Pico W

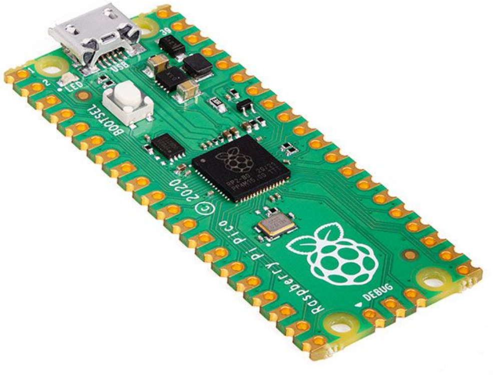
Raspberry Pi Pico is a low-cost, high-performance microcontroller board with flexible digital interfaces. Key features include:

- 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

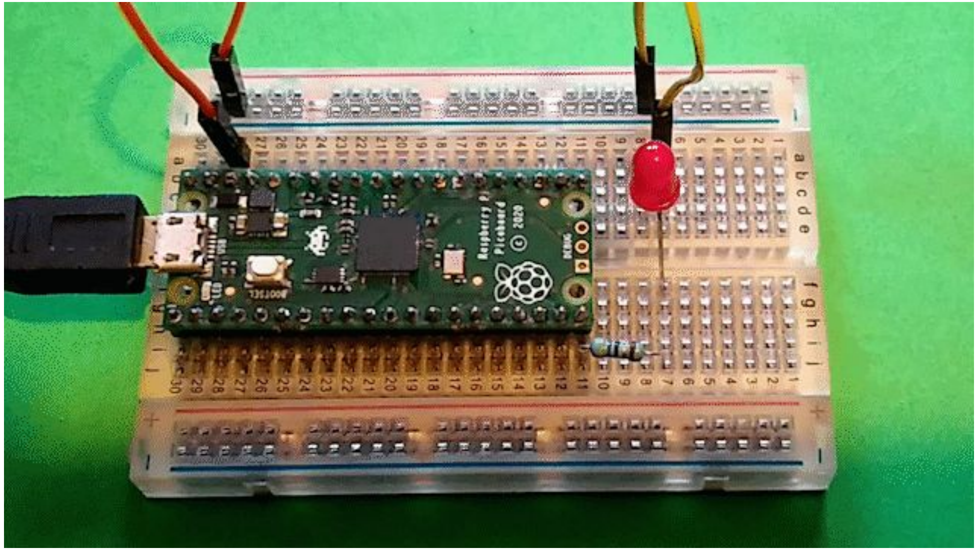
$CA8.34 at canada.newark.com

Images from digikey.ca

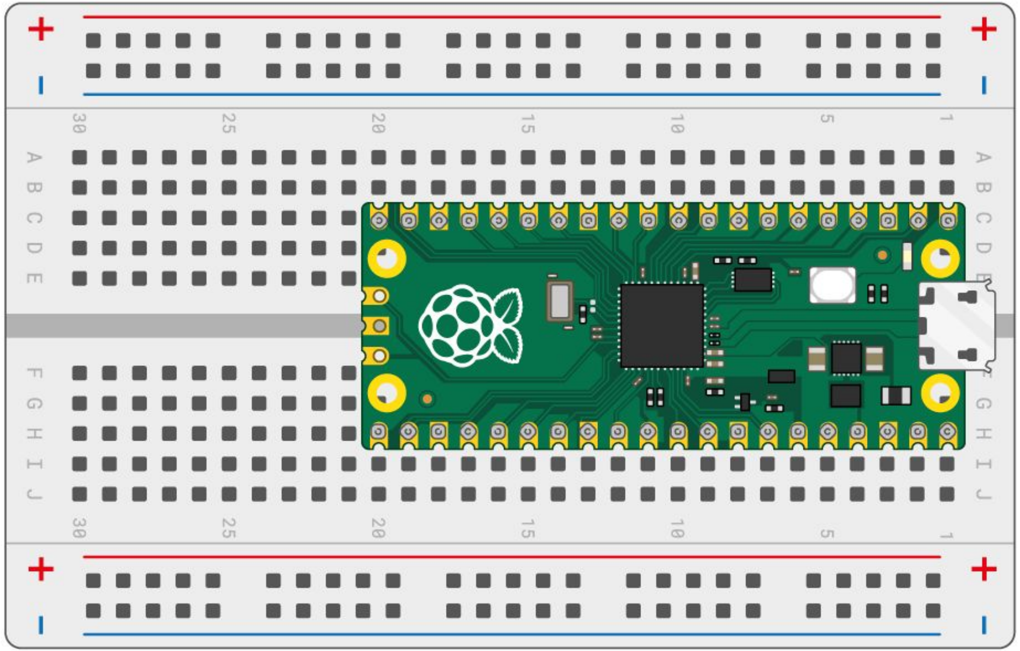
# Getting started with the Pico - MicroPython

The main reference is a very good tutorial

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

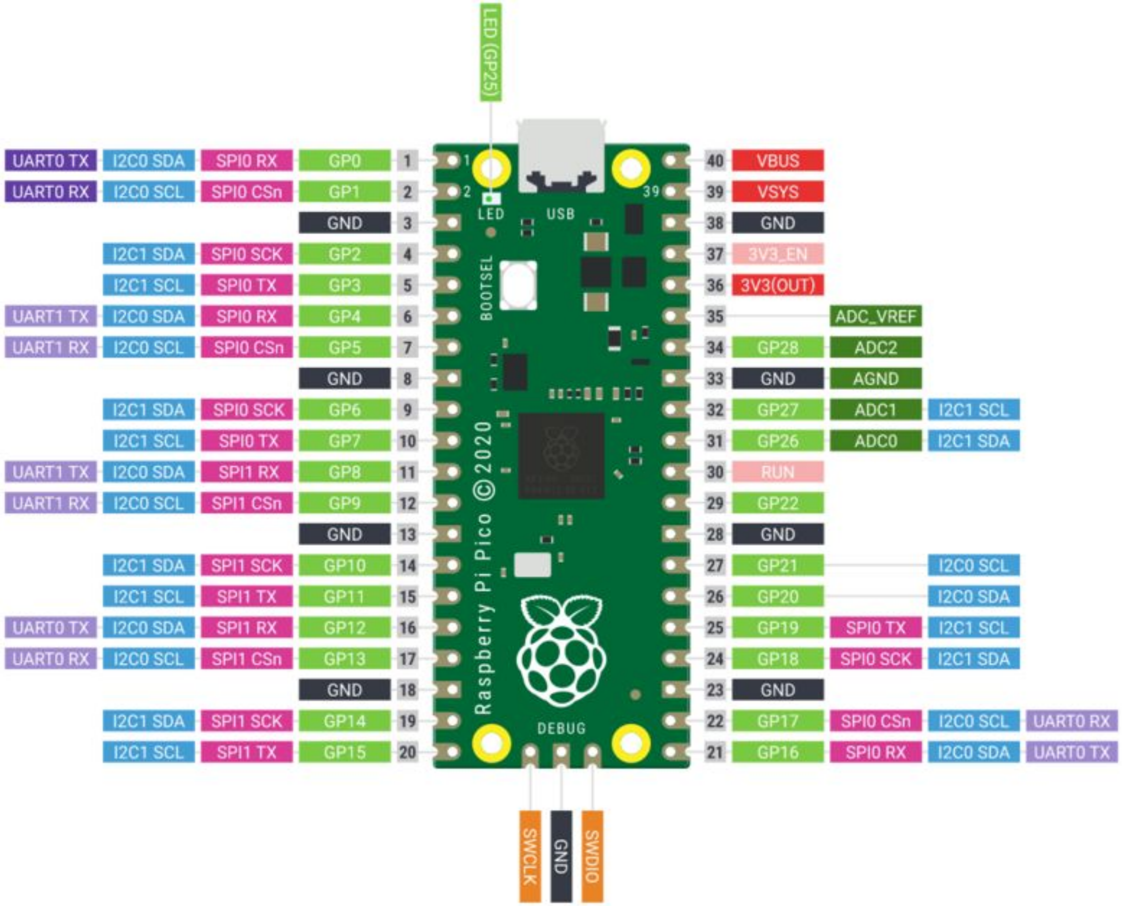


# Pico on breadboard



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

# RPi Pico Pins

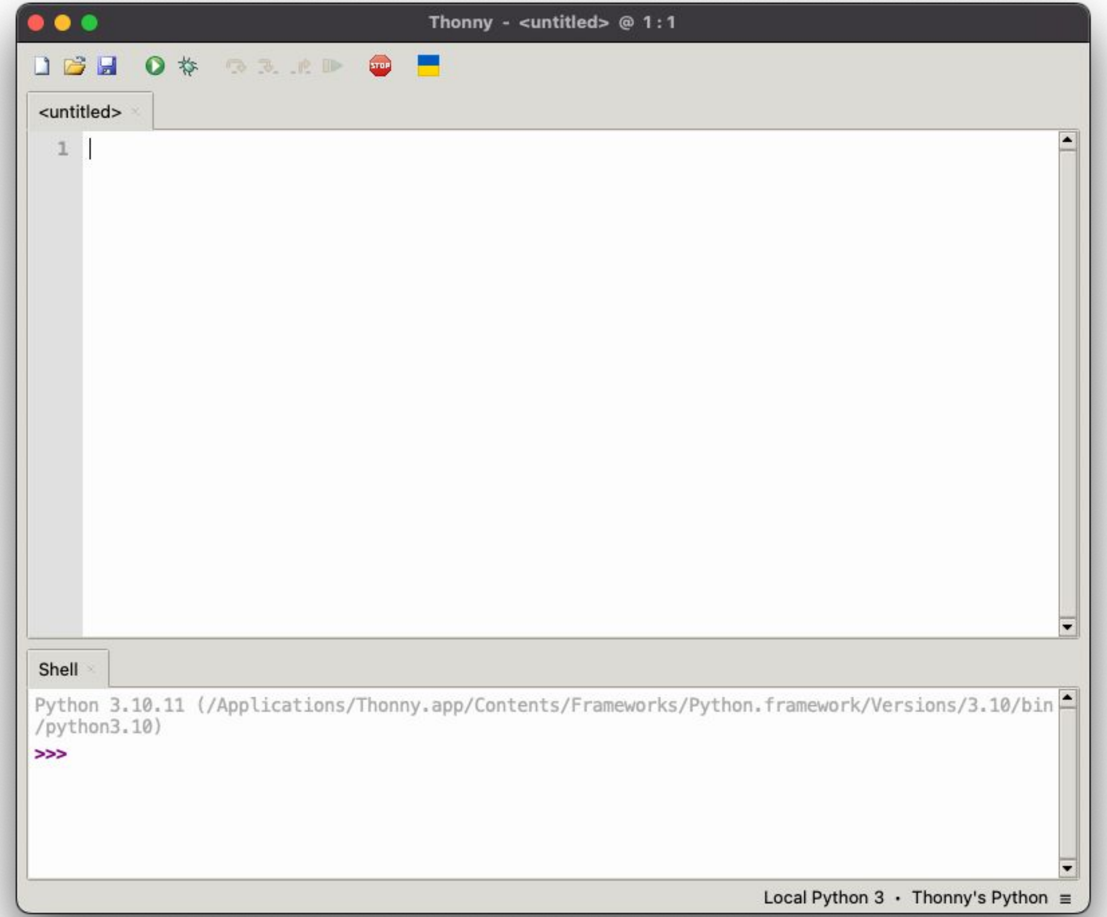


| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1 | 40 | VBUS | | | |
| UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2 | 39 | VSYS | | | |
| | | | GND | 3 | 38 | GND | | | |
| | I2C1 SDA | SPI0 SCK | GP2 | 4 | 37 | 3V3_EN | | | |
| | I2C1 SCL | SPI0 TX | GP3 | 5 | 36 | 3V3(OUT) | | | |
| UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6 | 35 | | ADC_VREF | | |
| UART1 RX | I2C0 SCL | SPI0 CSn | GP5 | 7 | 34 | GP28 | ADC2 | | |
| | | | GND | 8 | 33 | GND | AGND | | |
| | I2C1 SDA | SPI0 SCK | GP6 | 9 | 32 | GP27 | ADC1 | I2C1 SCL | |
| | I2C1 SCL | SPI0 TX | GP7 | 10 | 31 | GP26 | ADC0 | I2C1 SDA | |
| UART1 TX | I2C0 SDA | SPI1 RX | GP8 | 11 | 30 | RUN | | | |
| UART1 RX | I2C0 SCL | SPI1 CSn | GP9 | 12 | 29 | GP22 | | | |
| | | | GND | 13 | 28 | GND | | | |
| | I2C1 SDA | SPI1 SCK | GP10 | 14 | 27 | GP21 | | I2C0 SCL | |
| | I2C1 SCL | SPI1 TX | GP11 | 15 | 26 | GP20 | | I2C0 SDA | |
| UART0 TX | I2C0 SDA | SPI1 RX | GP12 | 16 | 25 | GP19 | SPI0 TX | I2C1 SCL | |
| UART0 RX | I2C0 SCL | SPI1 CSn | GP13 | 17 | 24 | GP18 | SPI0 SCK | I2C1 SDA | |
| | | | GND | 18 | 23 | GND | | | |
| | I2C1 SDA | SPI1 SCK | GP14 | 19 | 22 | GP17 | SPI0 CSn | I2C0 SCL | UART0 RX |
| | I2C1 SCL | SPI1 TX | GP15 | 20 | 21 | GP16 | SPI0 RX | I2C0 SDA | UART0 TX |

🟥 Power ⬛ Ground ◪ UART / UART (default) 🟩 GPIO, PIO, and PWM

🟩 ADC 🟪 SPI 🟦 I2C 🟥 System Control 🟧 Debugging

# Install Thonny (Win/Lin/macOS)



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

# Pico in boot select (bootsel) mode

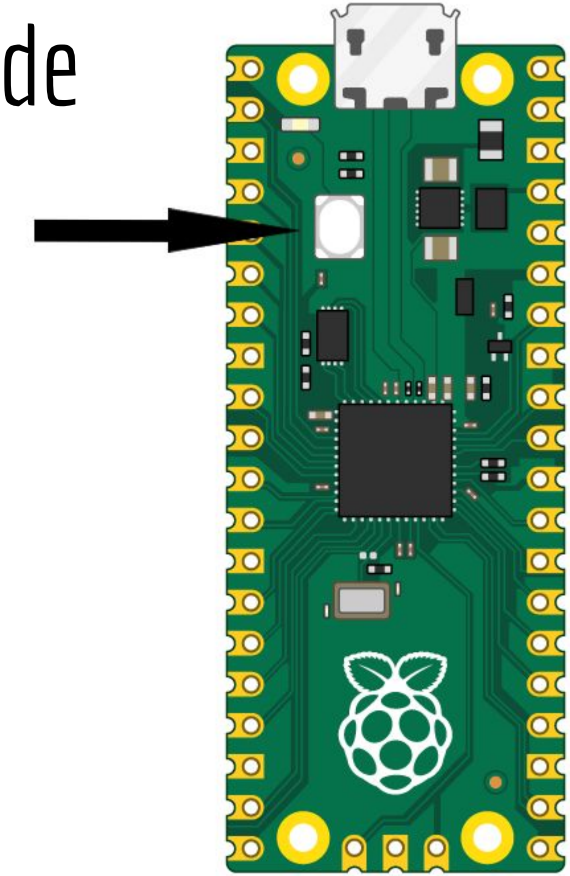
Unplug Pico from USB

Press and hold bootsel button

Plug Pico into USB and wait a few seconds

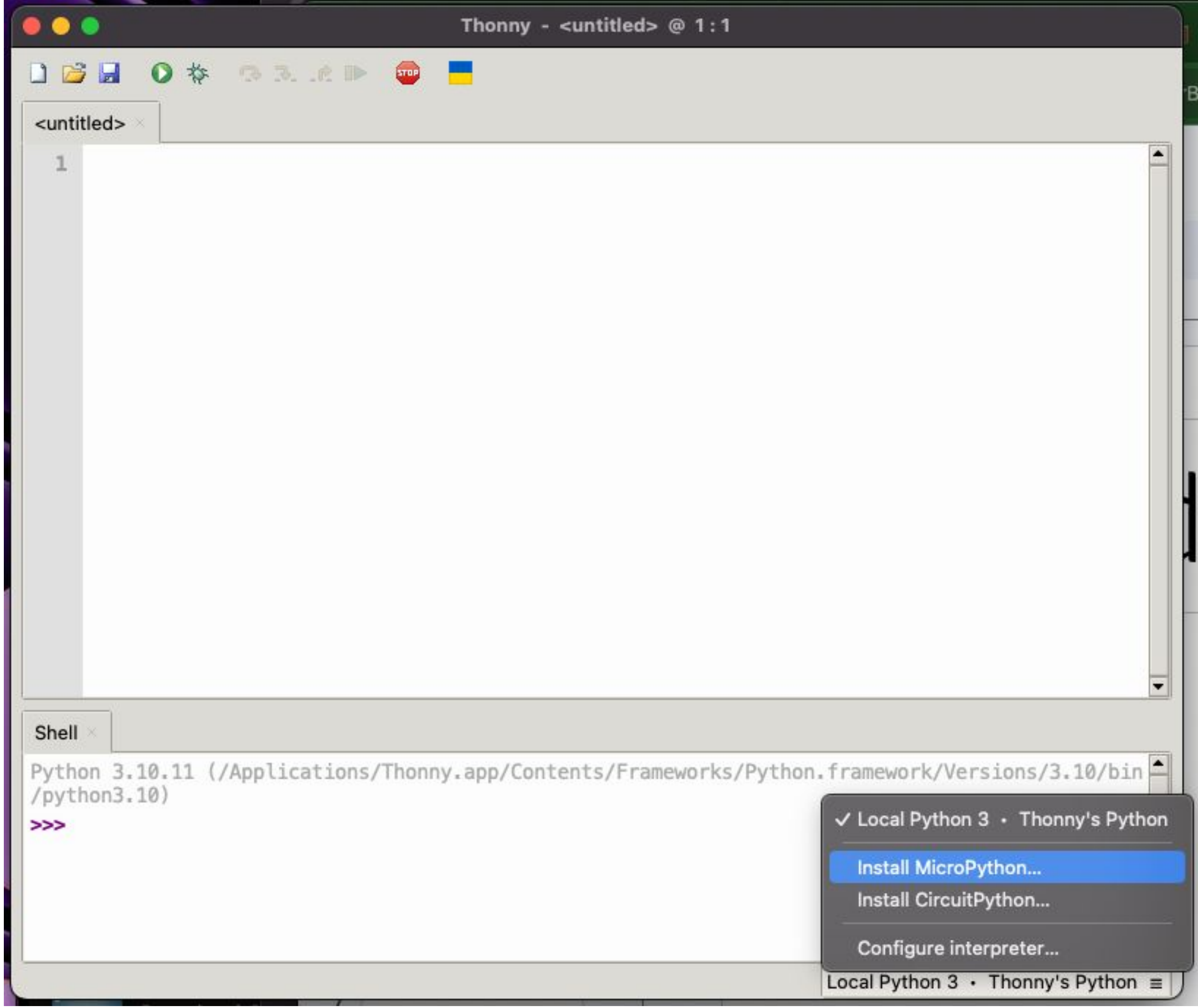
Release bootsel button

Open Thonny



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

# Install MicroPython

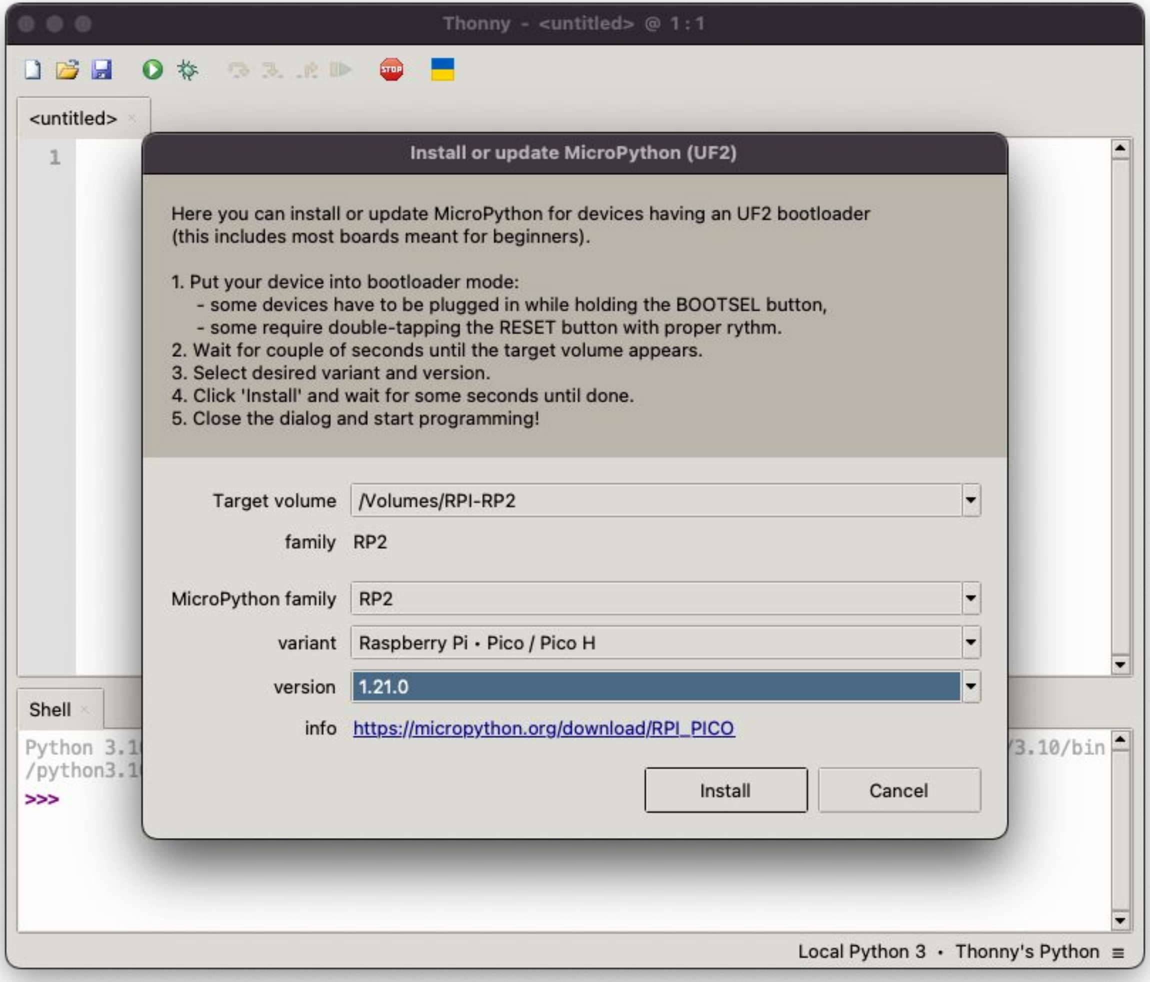


Thonny - <untitled> @ 1 : 1
<untitled>
1
Shell

```
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 ≡

# 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

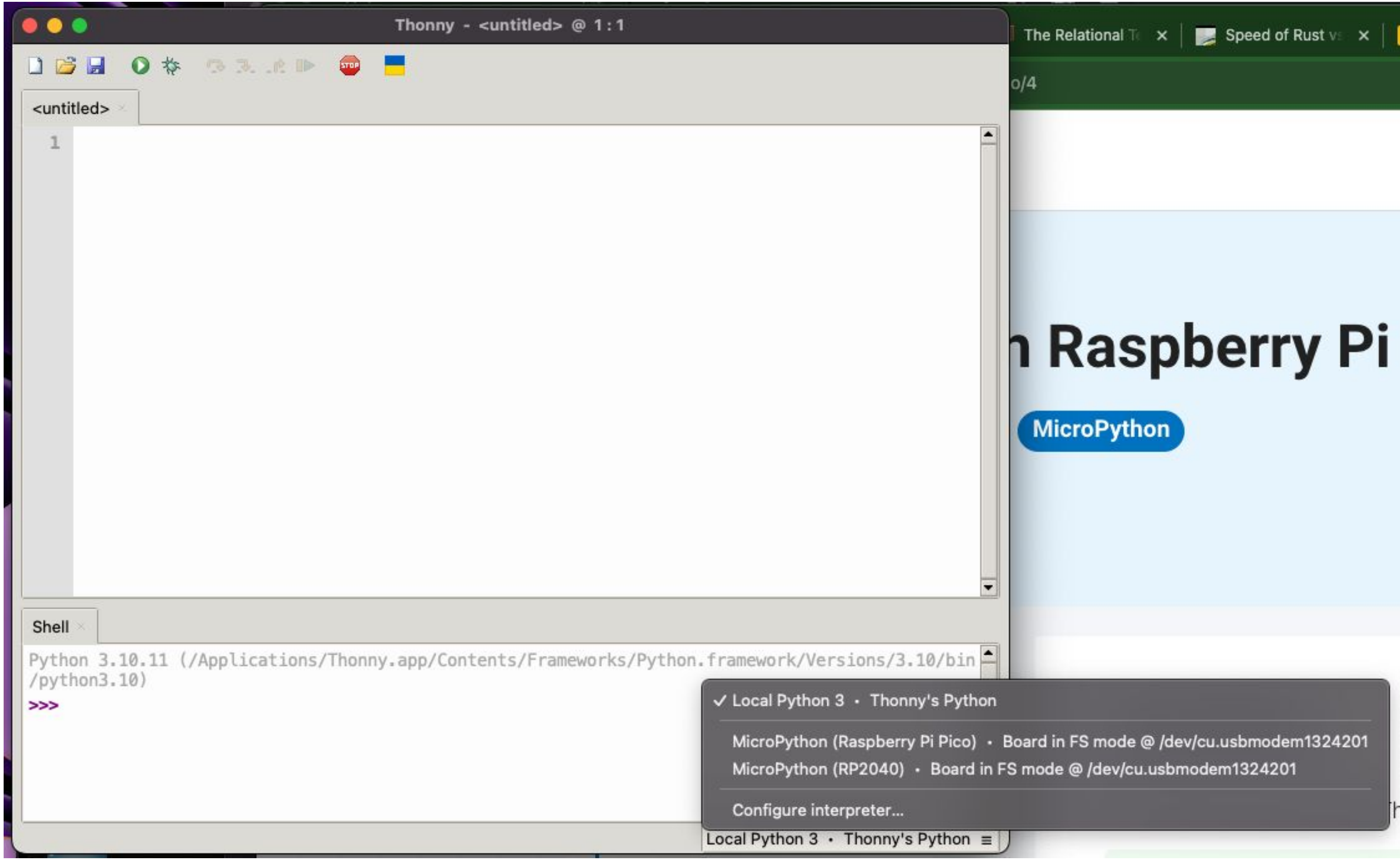
MicroPython family: RP2

variant: Raspberry Pi • Pico / Pico H

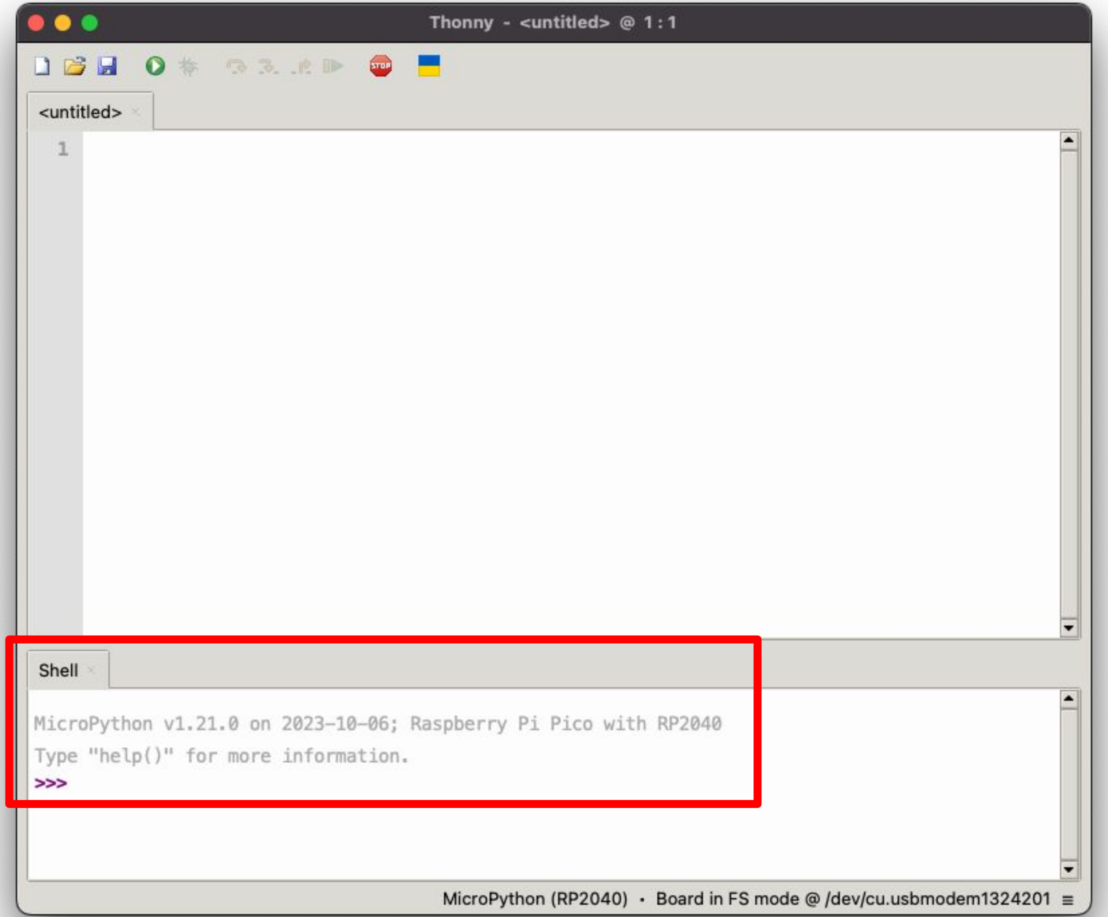
version: 1.21.0

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

# Connect


Thonny - <untitled> @ 1 : 1
The Relational T
Speed of Rust v
o/4
<untitled>
1
n Raspberry Pi
MicroPython
Shell
✓ Local Python 3 • Thonny's Python
MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/cu.usbmodem1324201
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324201
Configure interpreter...
Local Python 3 • Thonny's Python ≡

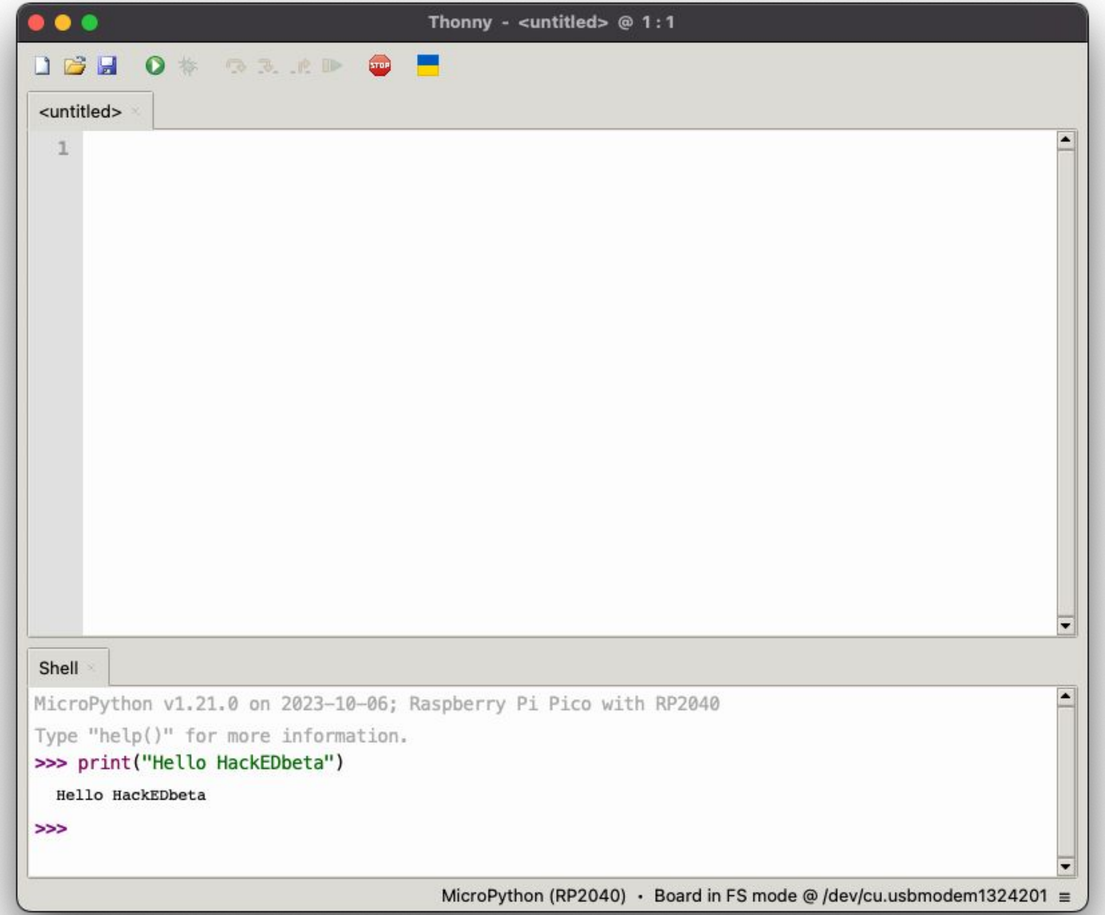
```
Python 3.10.11 (/Applications/Thonny.app/Contents/Frameworks/Python.framework/Versions/3.10/bin
/python3.10)
>>>
```



```
MicroPython v1.21.0 on 2023-10-06; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324201

# Success!



```
MicroPython v1.21.0 on 2023-10-06; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> print("Hello HackEDbeta")
 Hello HackEDbeta
>>>
```

MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324201

# GPIOs

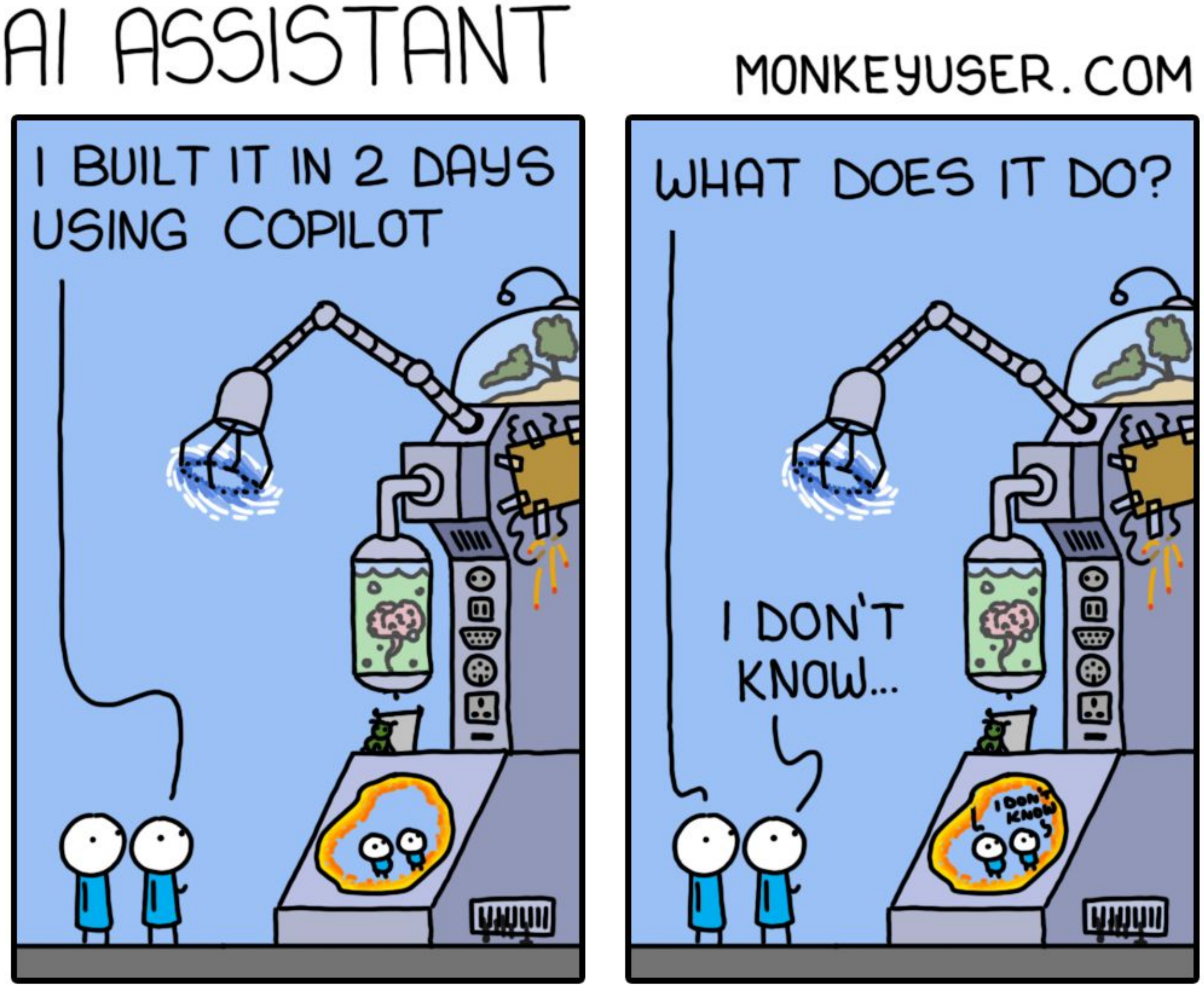
Lots of support for GPIO and other peripherals at MicroPython.org

Quick Reference for MicroPython is found at

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

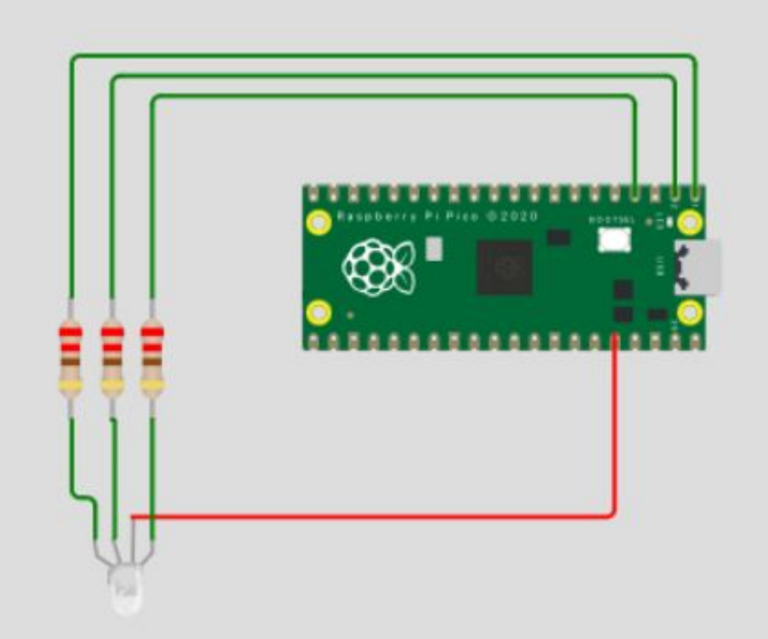
Support for

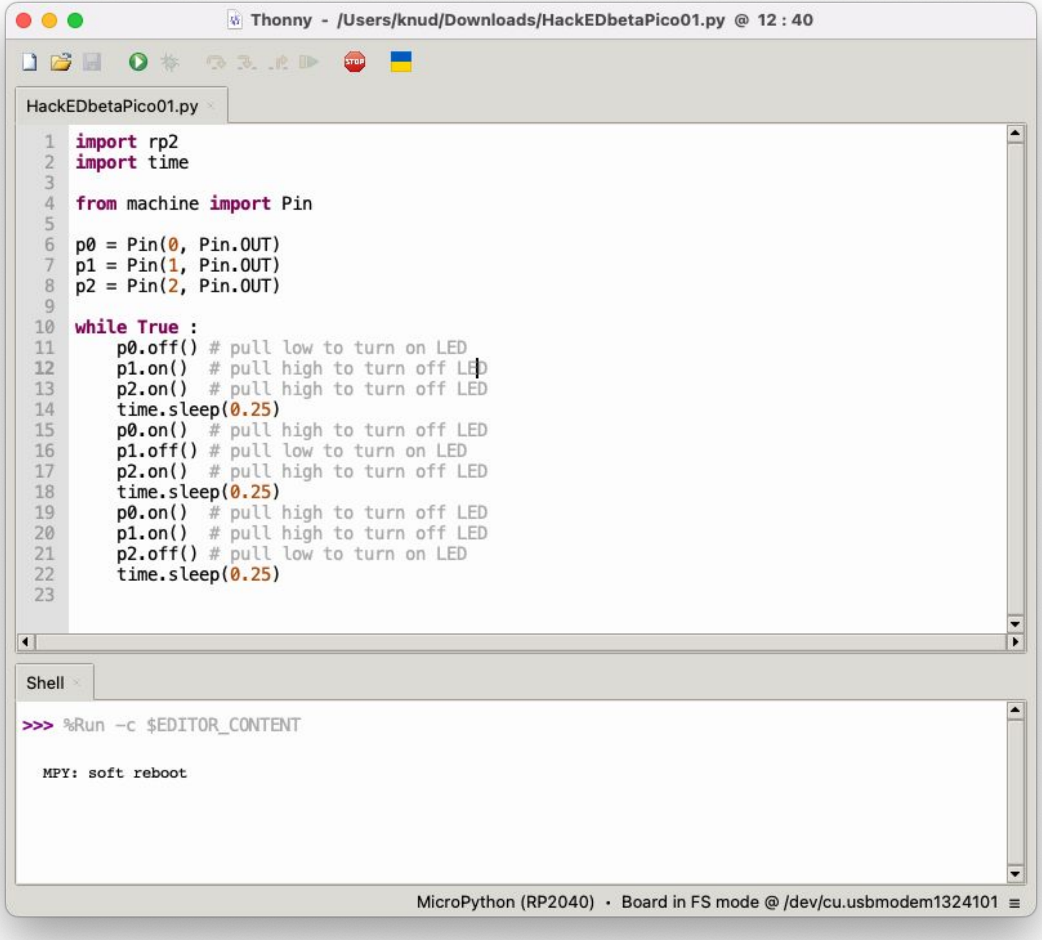
- HW Timers, GPIO, UART, PWM, ADC, SW SPI, HW SPI, SW I2C, HW I2C, I2S, RTC, WDT, OneWire, NeoPixel, APA106
- … but probably not …



monkeyuser.com

# Simple RGB blinky




Thonny - /Users/knud/Downloads/HackEDbetaPico01.py @ 12 : 40
HackEDbetaPico01.py
Shell
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324101

```
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
```

# 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 23.04 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)

# Follow the Getting Started

Follow the Getting Started Guide…

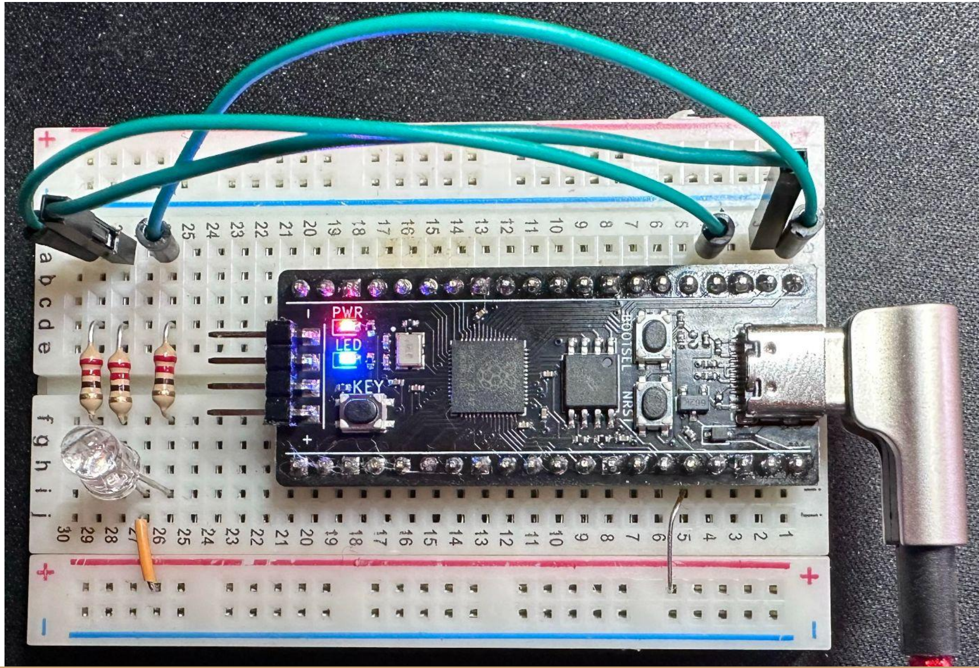
Do the “blink” example to be sure the tool chain is up and running.

For this example I use an RP2040 board from AliExpress.

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

Different Pico

From Universal Solder



# 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/pico-examples
cp -dpR blink 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 `blinky` example is added
    b. Replicate those lines and replace `blinky` 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

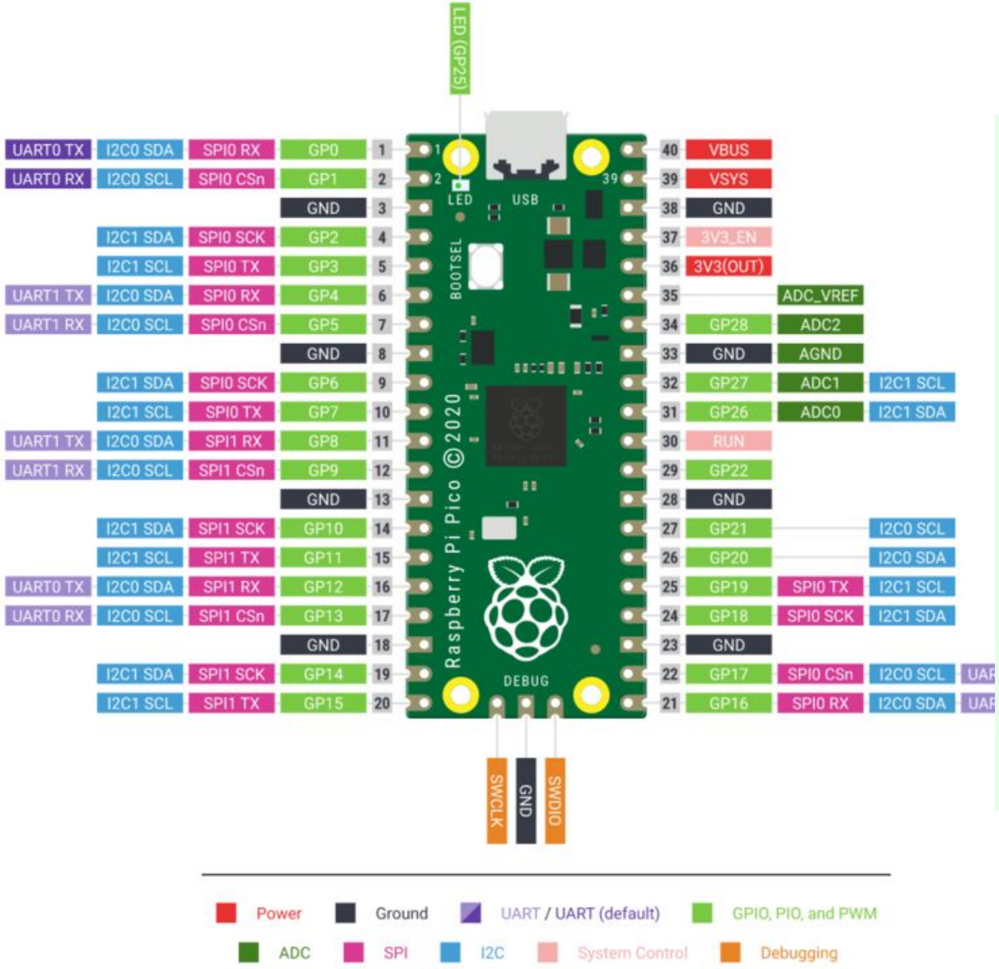
    ```
    make -j4
    ```

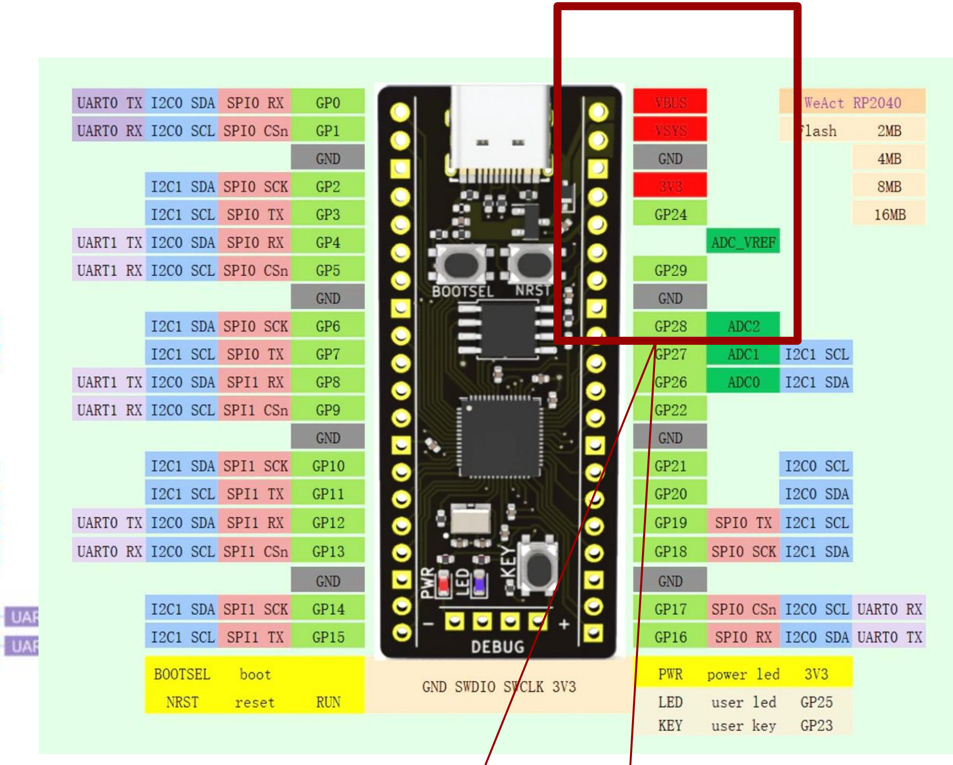
6. There should be a new executable named `rgb_demo`.
7. Put the Pico back into bootloader mode and drop the `rgb_demo.uf2` file on the drive – the old blink demo should work

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);
    }
}
```

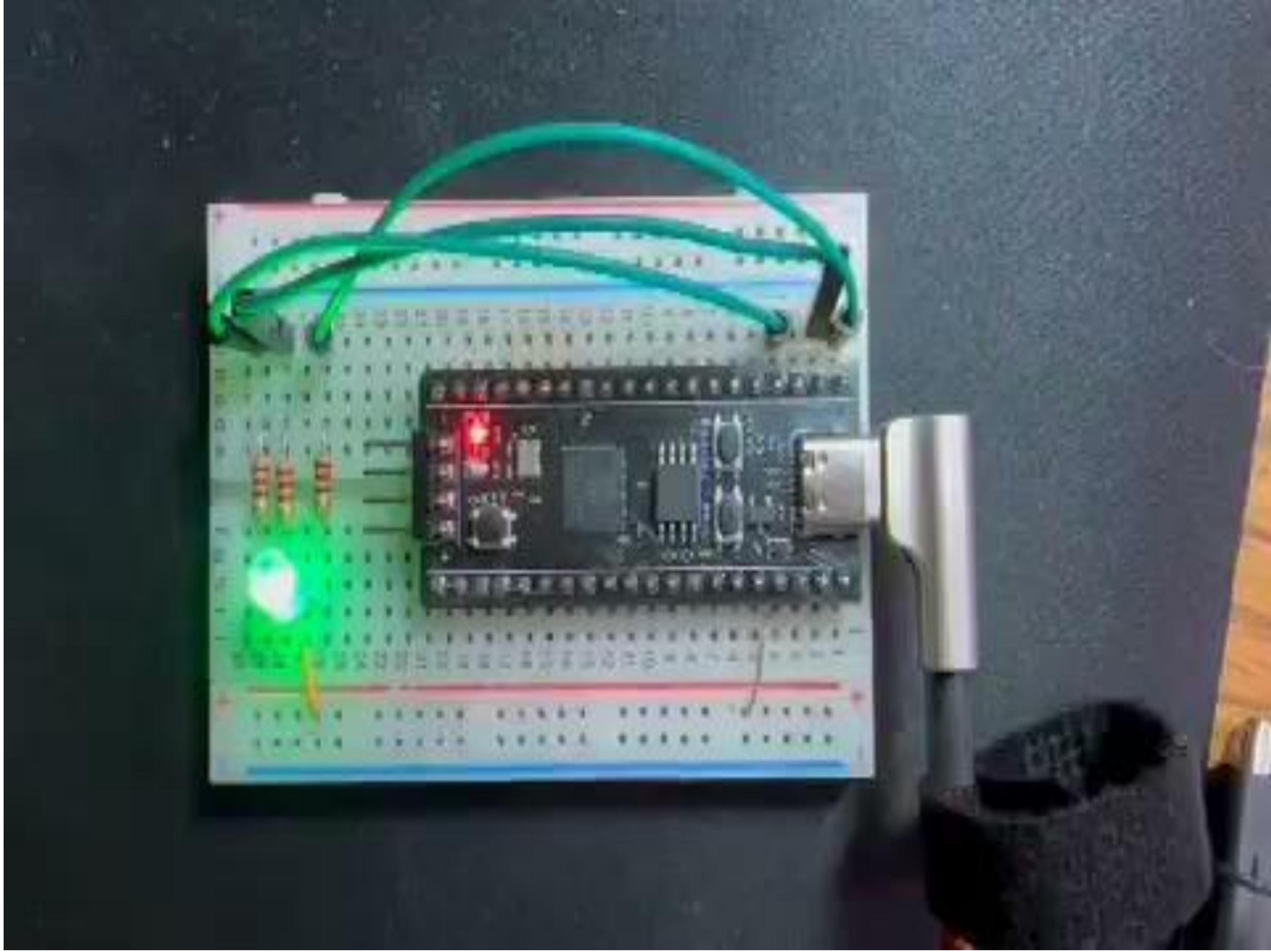
# GOTCHA!


LED (GP25)
UART0 TX I2C0 SDA SPI0 RX GP0 1
UART0 RX I2C0 SCL SPI0 CSn GP1 2
GND 3
I2C1 SDA SPI0 SCK GP2 4
I2C1 SCL SPI0 TX GP3 5
UART1 TX I2C0 SDA SPI0 RX GP4 6
UART1 RX I2C0 SCL SPI0 CSn GP5 7
GND 8
I2C1 SDA SPI0 SCK GP6 9
I2C1 SCL SPI0 TX GP7 10
UART1 TX I2C0 SDA SPI1 RX GP8 11
UART1 RX I2C0 SCL SPI1 CSn GP9 12
GND 13
I2C1 SDA SPI1 SCK GP10 14
I2C1 SCL SPI1 TX GP11 15
UART0 TX I2C0 SDA SPI1 RX GP12 16
UART0 RX I2C0 SCL SPI1 CSn GP13 17
GND 18
I2C1 SDA SPI1 SCK GP14 19
I2C1 SCL SPI1 TX GP15 20
40 VBUS
39 VSYS
38 GND
37 3V3_EN
36 3V3(OUT)
35 ADC_VREF
34 GP28 ADC2
33 GND AGND
32 GP27 ADC1 I2C1 SCL
31 GP26 ADC0 I2C1 SDA
30 RUN
29 GP22
28 GND
27 GP21 I2C0 SCL
26 GP20 I2C0 SDA
25 GP19 SPI0 TX I2C1 SCL
24 GP18 SPI0 SCK I2C1 SDA
23 GND
22 GP17 SPI0 CSn I2C0 SCL UAR
21 GP16 SPI0 RX I2C0 SDA UAR
1
2
39
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020
DEBUG
SWCLK
GND
SWDIO
Power
Ground
UART / UART (default)
GPIO, PIO, and PWM
ADC
SPI
I2C
System Control
Debugging


UART0 TX I2C0 SDA SPI0 RX GP0
UART0 RX I2C0 SCL SPI0 CSn GP1
GND
I2C1 SDA SPI0 SCK GP2
I2C1 SCL SPI0 TX GP3
UART1 TX I2C0 SDA SPI0 RX GP4
UART1 RX I2C0 SCL SPI0 CSn GP5
GND
I2C1 SDA SPI0 SCK GP6
I2C1 SCL SPI0 TX GP7
UART1 TX I2C0 SDA SPI1 RX GP8
UART1 RX I2C0 SCL SPI1 CSn GP9
GND
I2C1 SDA SPI1 SCK GP10
I2C1 SCL SPI1 TX GP11
UART0 TX I2C0 SDA SPI1 RX GP12
UART0 RX I2C0 SCL SPI1 CSn GP13
GND
I2C1 SDA SPI1 SCK GP14
I2C1 SCL SPI1 TX GP15
VBUS
VSYS
GND
3V3
GP24
ADC_VREF
GP29
GND
GP28 ADC2
GP27 ADC1 I2C1 SCL
GP26 ADC0 I2C1 SDA
GP22
GND
GP21 I2C0 SCL
GP20 I2C0 SDA
GP19 SPI0 TX I2C1 SCL
GP18 SPI0 SCK I2C1 SDA
GND
GP17 SPI0 CSn I2C0 SCL UART0 RX
GP16 SPI0 RX I2C0 SDA UART0 TX
WeAct RP2040
Flash 2MB
4MB
8MB
16MB
BOOTSEL
NRST
PWR
LED
KEY
DEBUG
BOOTSEL boot
NRST reset RUN
GND SWDIO SWCLK 3V3
PWR power led 3V3
LED user led GP25
KEY user key GP23

**NOT THE SAME!**

# Live demo



# What about Rust

Doable

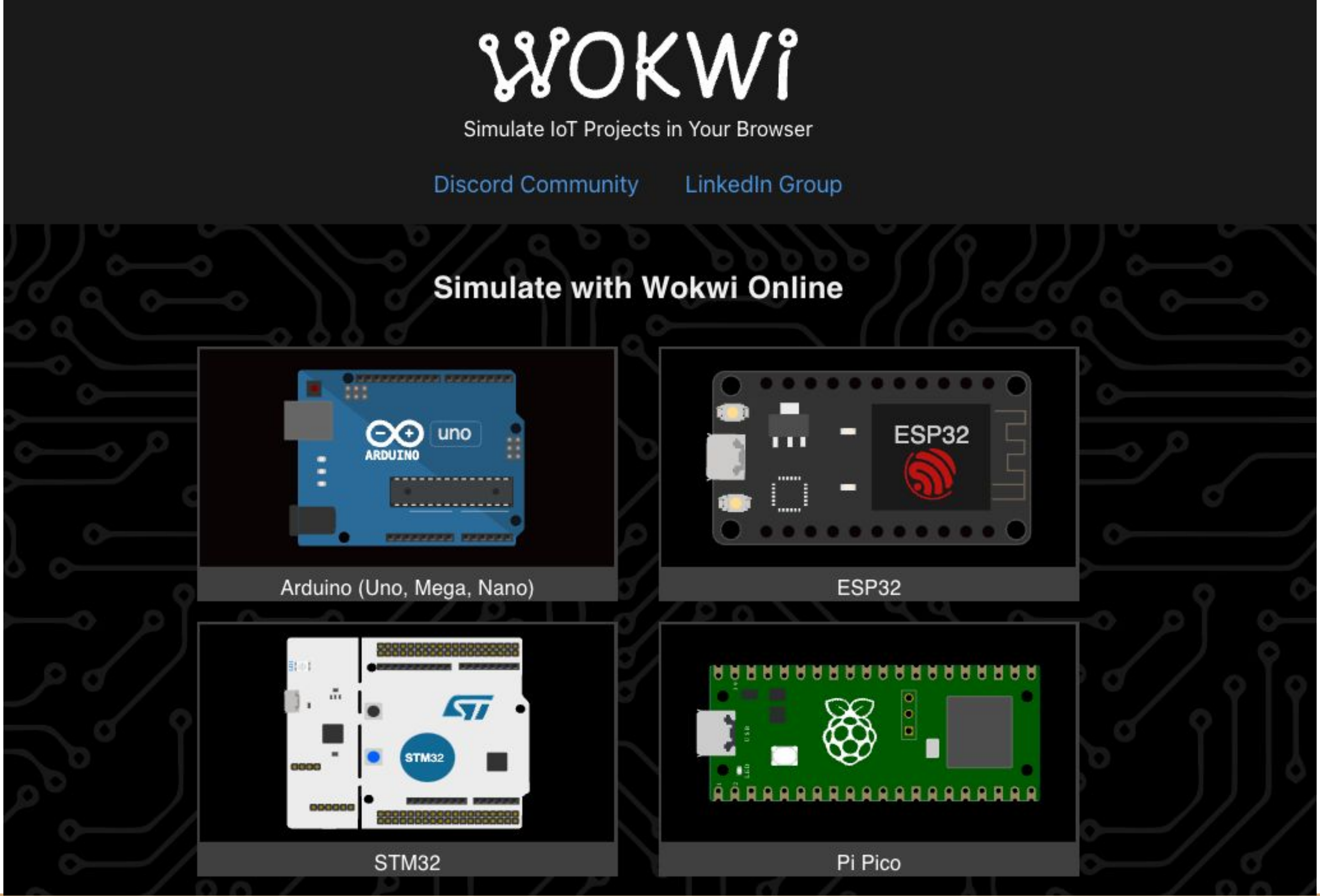
However, we don’t have time today to go through all the steps, it’s involved!

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

# Wokwi.com



WOKWI

Simulate IoT Projects in Your Browser

Discord Community LinkedIn Group

## Simulate with Wokwi Online

Arduino (Uno, Mega, Nano)

ESP32

STM32

Pi Pico

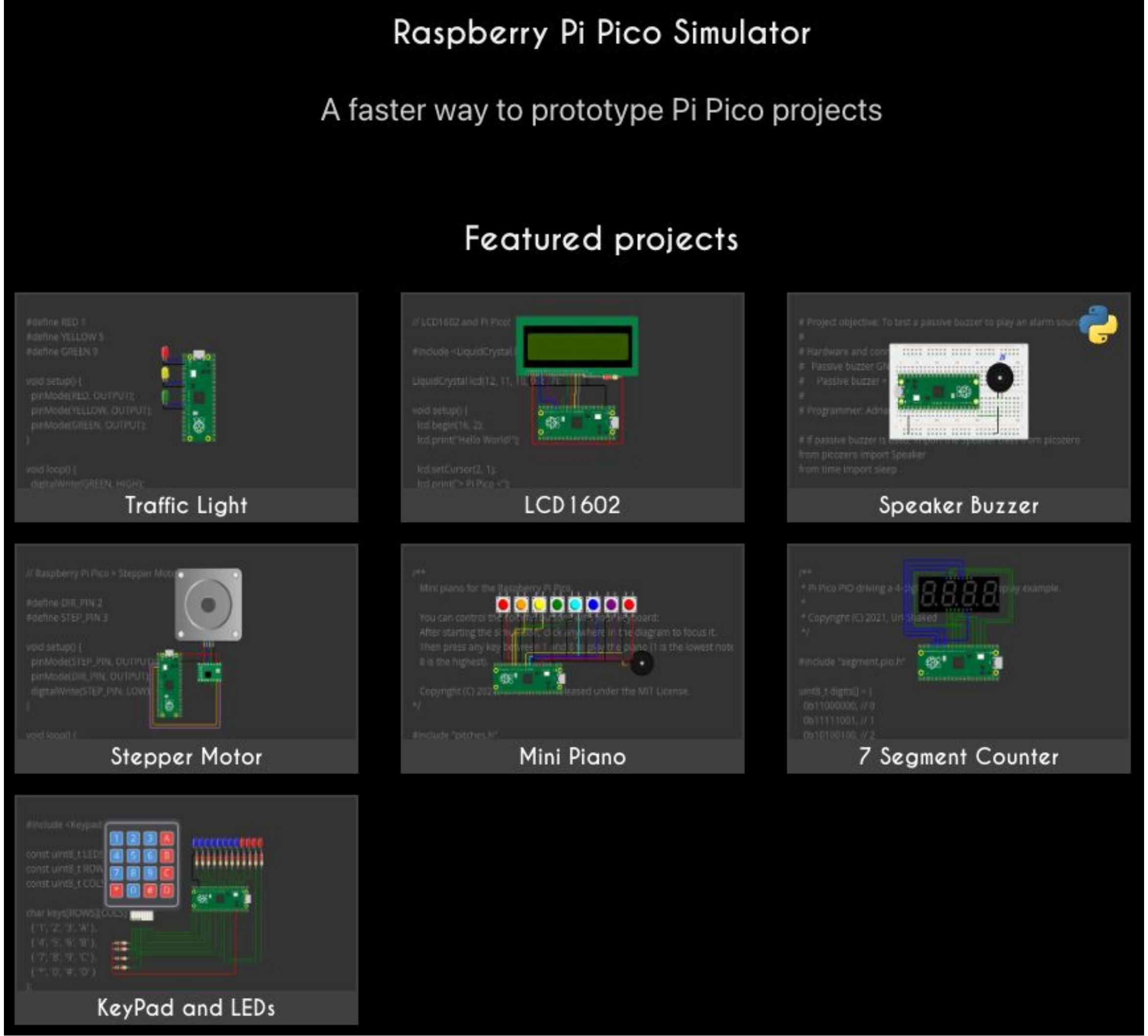
# What is Wokwi.com? (from docs.wokwi.com)

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

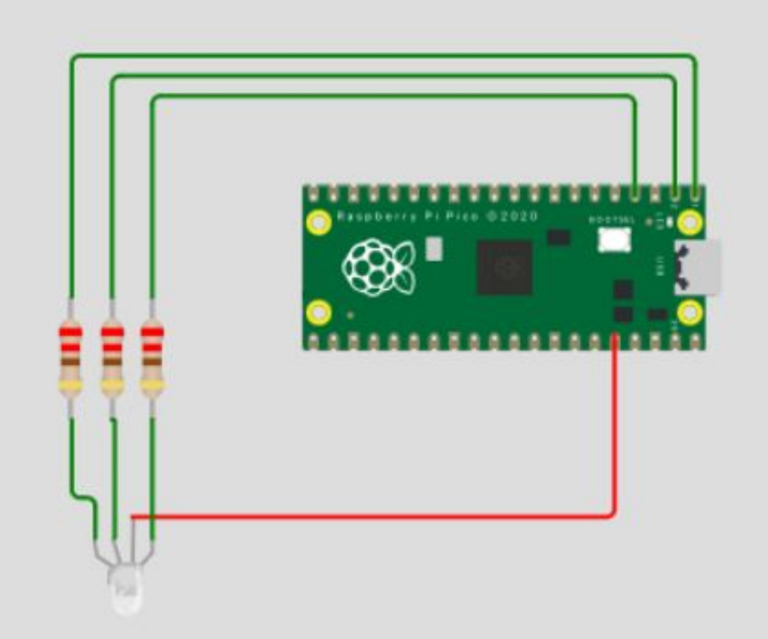
Stepper Motor

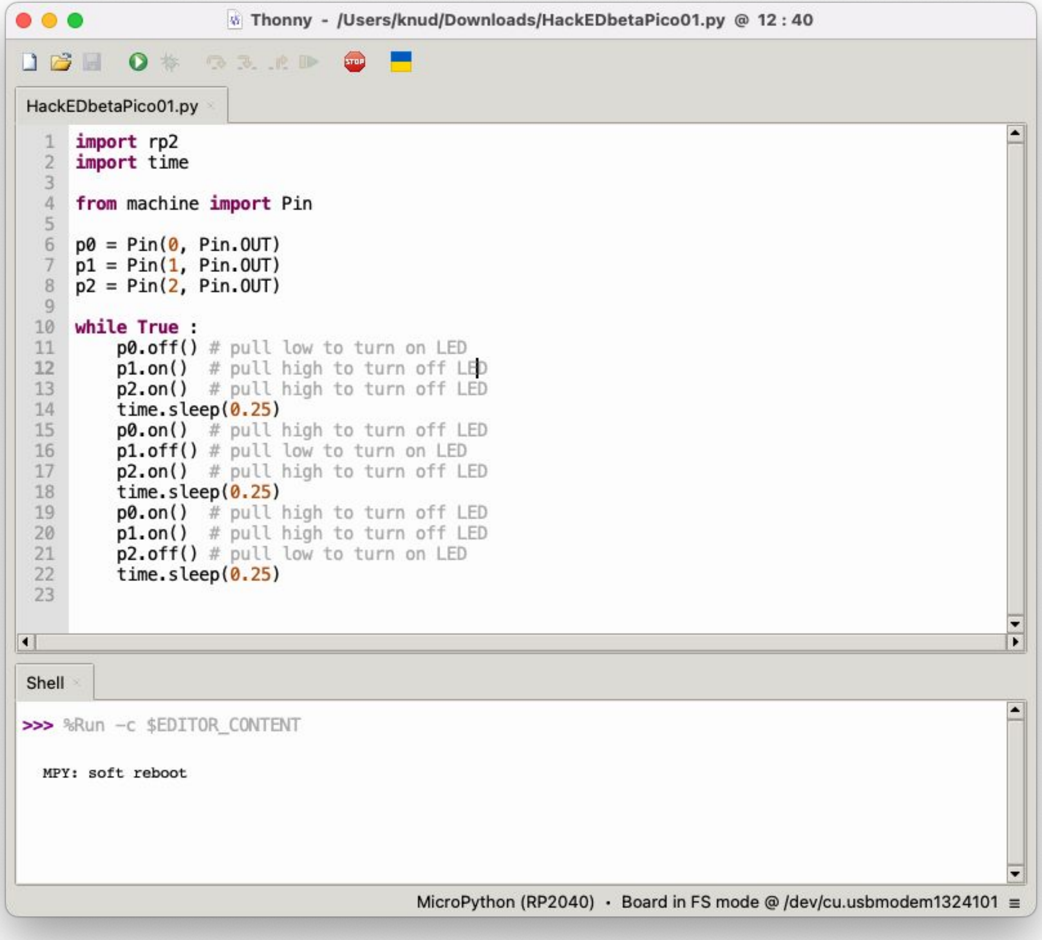
Mini Piano

7 Segment Counter

KeyPad and LEDs

# Simple RGB blinky

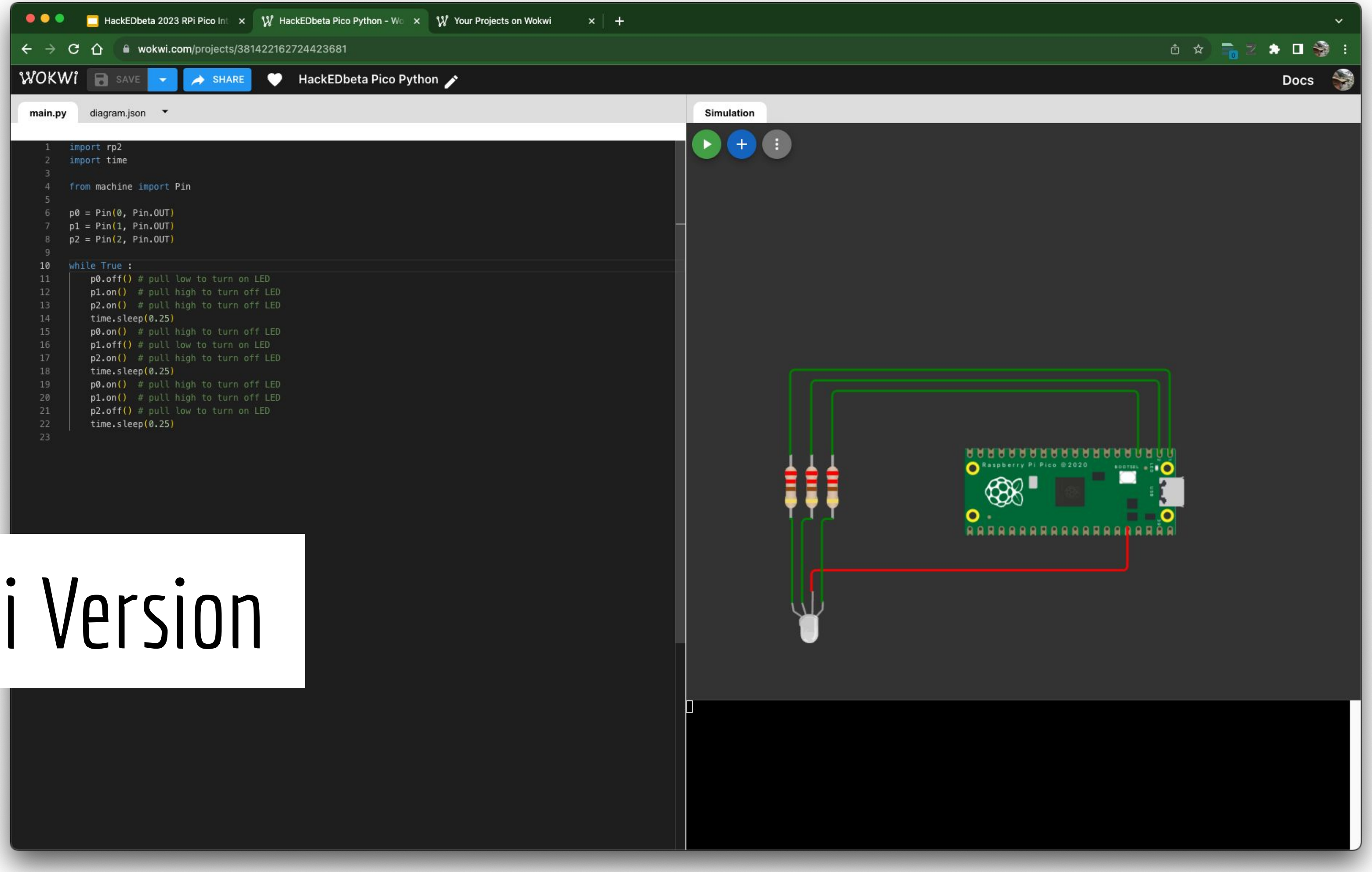



Thonny - /Users/knud/Downloads/HackEDbetaPico01.py @ 12 : 40
HackEDbetaPico01.py
Shell
MicroPython (RP2040) • Board in FS mode @ /dev/cu.usbmodem1324101

```python
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
```

# Wokwi Version



```python
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)

```

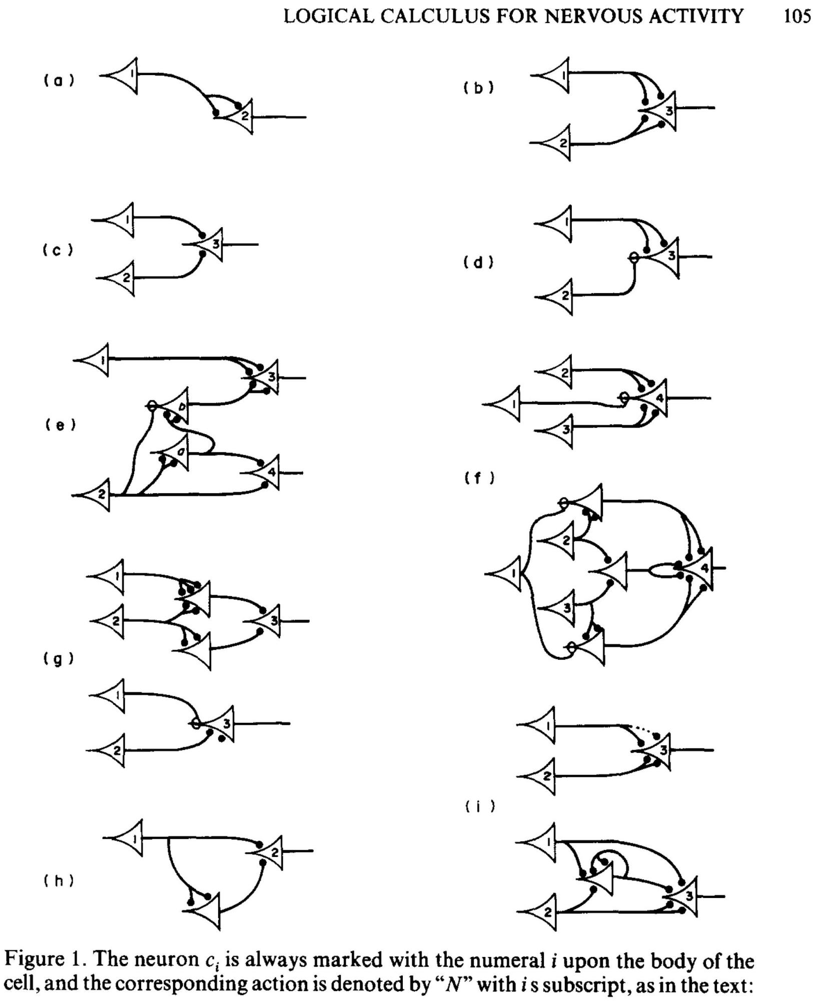
# Trivia

*...with wee prizes...*

# Trivia

Who created the first artificial neural network and when?

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



LOGICAL CALCULUS FOR NERVOUS ACTIVITY 105

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 $i$ s 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”

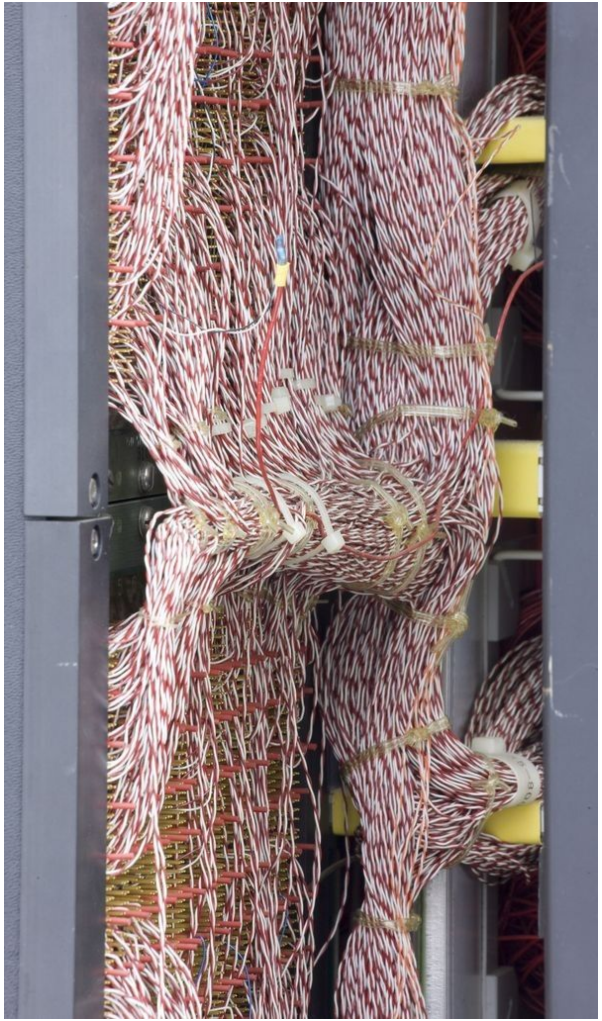
Keywords : Clones; Plug compat; IBM

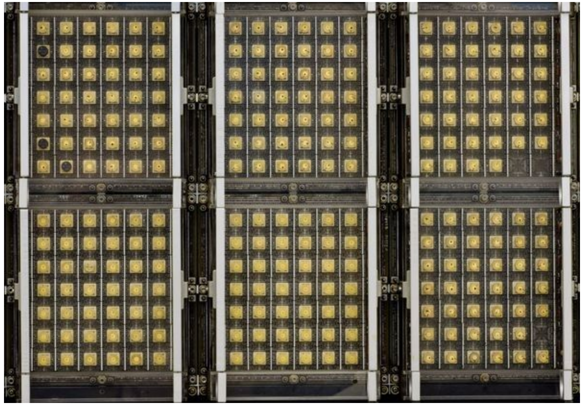
Speed : 3.5 MIPS

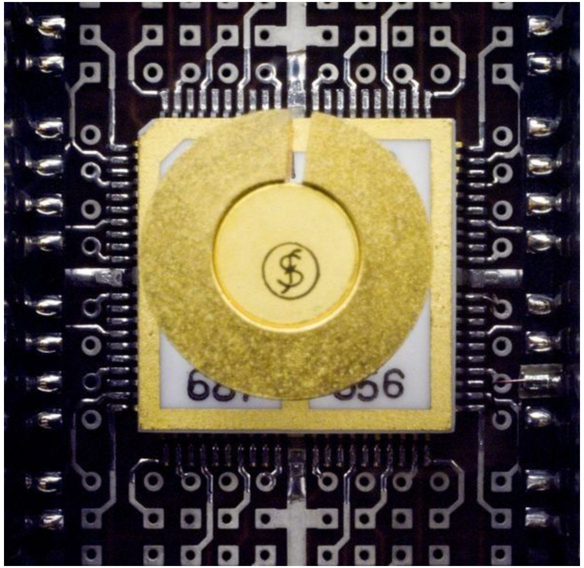
Memory Size : up to 8MB

Memory Width : 32-bit

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







# Arduino

Date Introduced : 2010

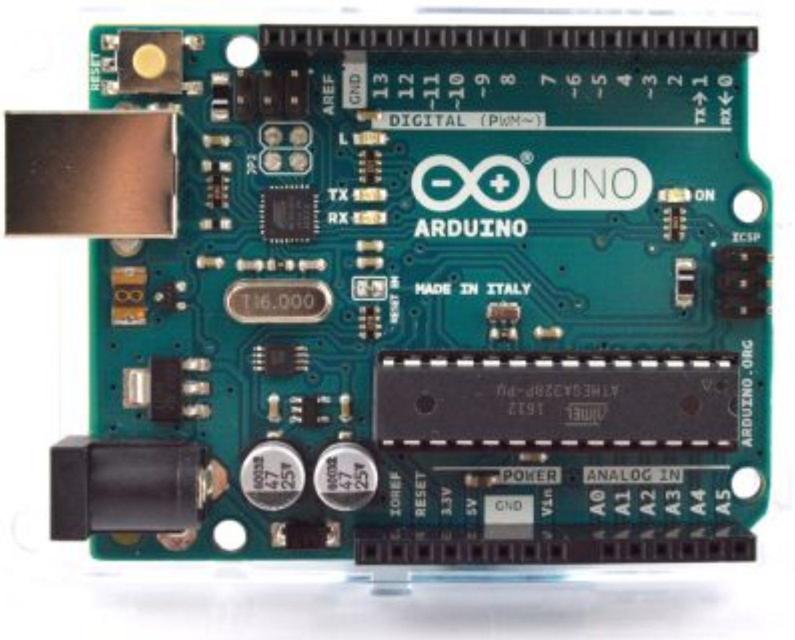
Dimensions overall: 2.7” x 2.1” x 0.6”

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%

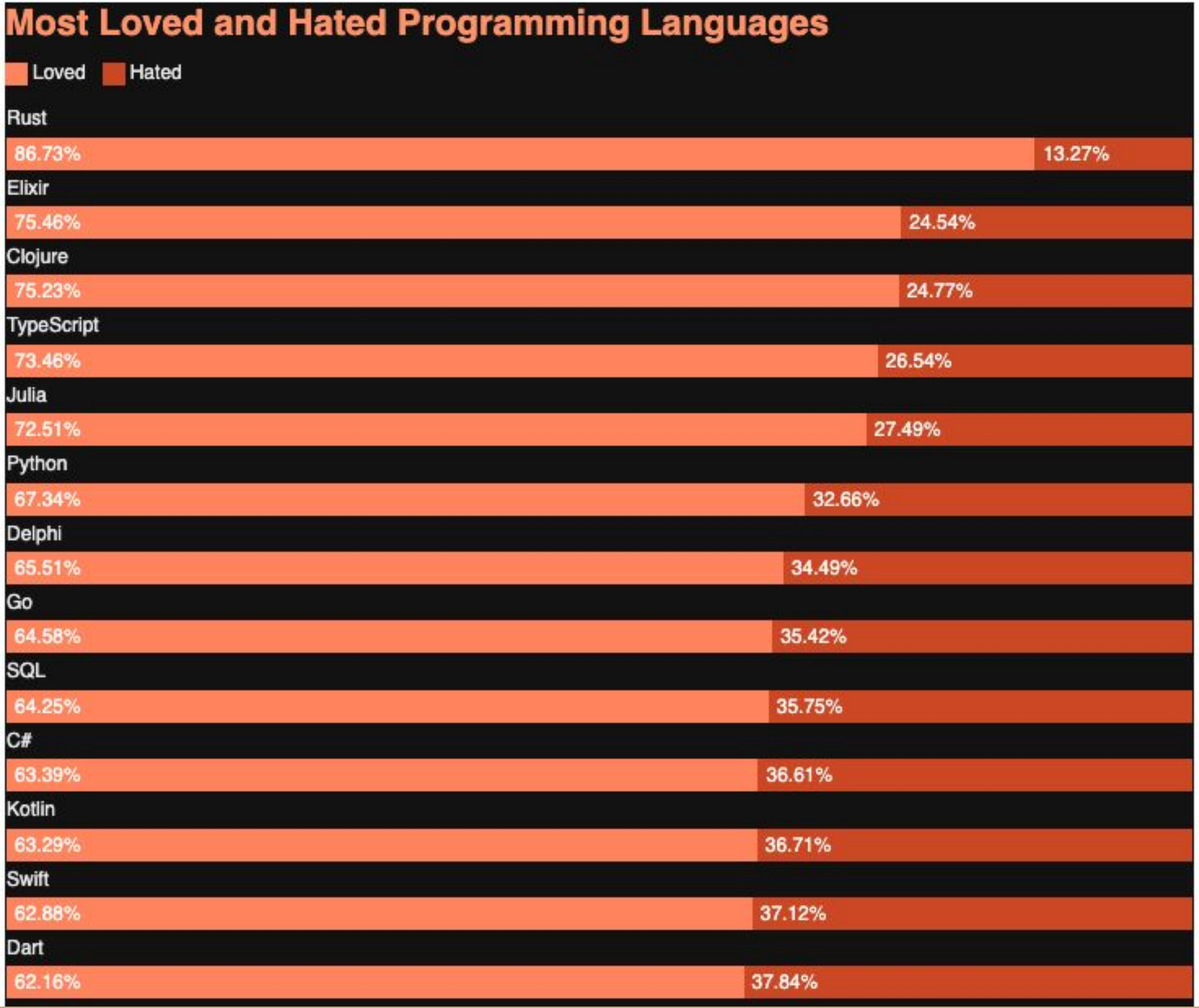
Elixir 75.46%

Clojure 75.23%

TypeScript 73.46%

Julia 72.51%

Python 67.34%

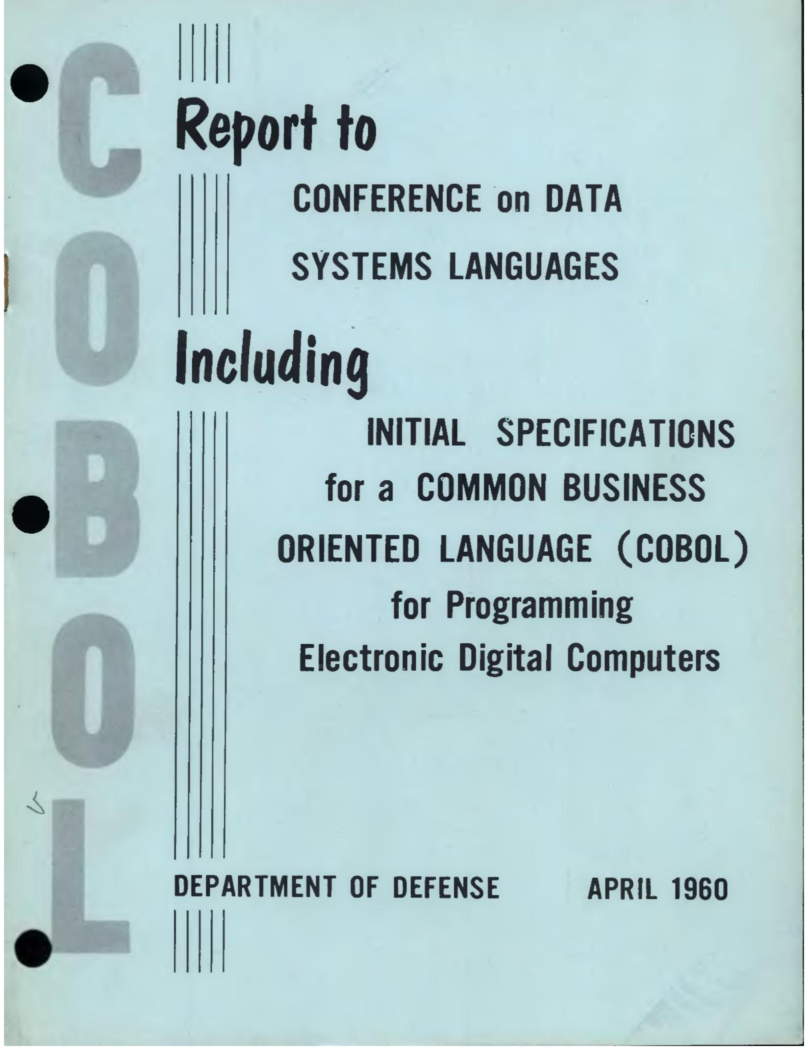


# Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and



COBOL

Report to

CONFERENCE on DATA
SYSTEMS LANGUAGES

Including

INITIAL SPECIFICATIONS
for a COMMON BUSINESS
ORIENTED LANGUAGE (COBOL)
for Programming
Electronic Digital Computers

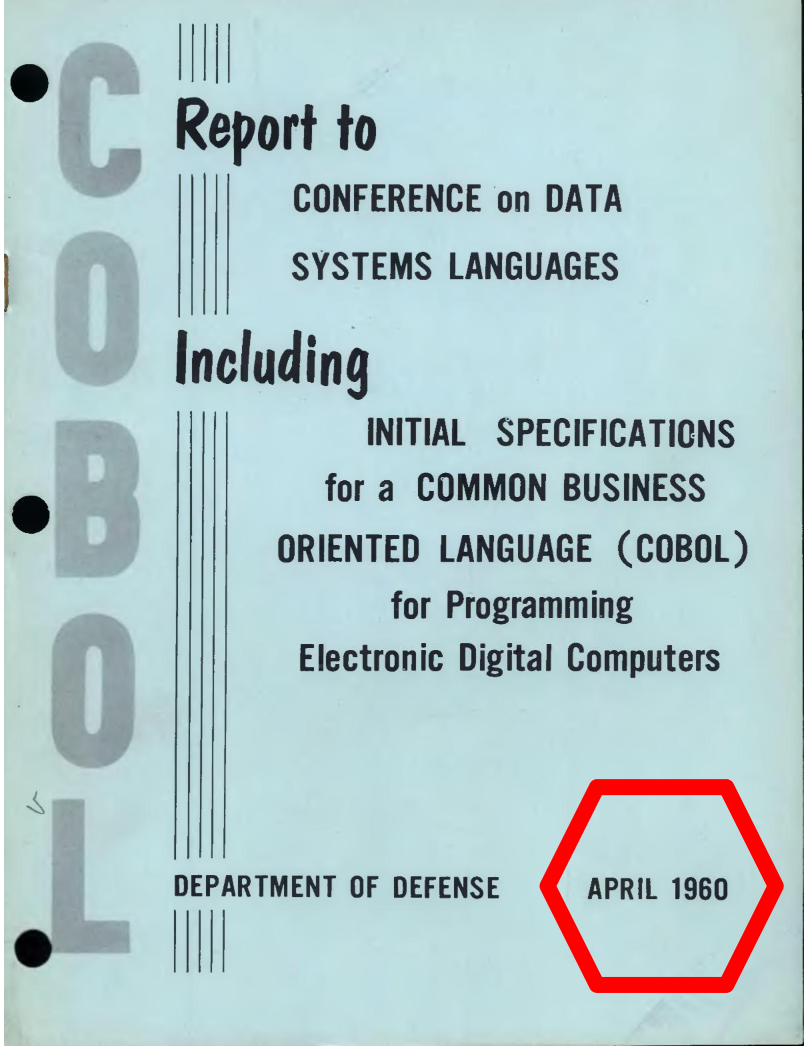
DEPARTMENT OF DEFENSE APRIL 1960

# Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and



COBOL

Report to

CONFERENCE on DATA
SYSTEMS LANGUAGES

Including

INITIAL SPECIFICATIONS
for a COMMON BUSINESS
ORIENTED LANGUAGE (COBOL)
for Programming
Electronic Digital Computers

DEPARTMENT OF DEFENSE

APRIL 1960

STACK OVERFLOW
YOU ARE BY FAR THE WORST PROGRAMMING LANGUAGE I HAVE EVER HEARD OF
VISUAL BASIC
W
X
A
BUT YOU HAVE HEARD OF ME
imgflip.com