HackED 2025
Raspberry Pi Pico with Rust

Steven Knudsen, PhD PEng
knud@ualberta.ca

DEPARTMENT OF

Electrical and Computer Engineering

ECE 421 - Exploring Software Development Domains

3 units (fi 8)(EITHER, 2-0-3)

Faculty of Engineering

Outline

1. Raspberry Pi Pico and Pico W overview
2. Why Rust?
3. Setting up for Rust development
4. Setting up Rust on a Pico
5. Simple Program
6. More complicate Program
7. Wokwi.com and Pico

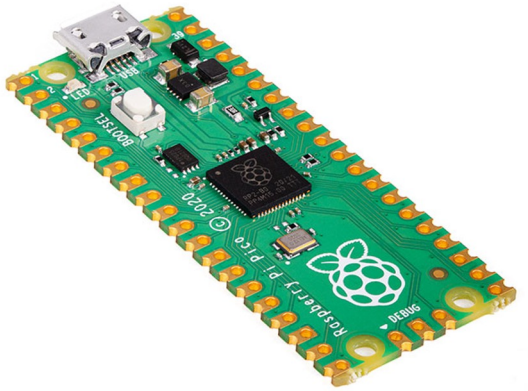
Sources for the talk

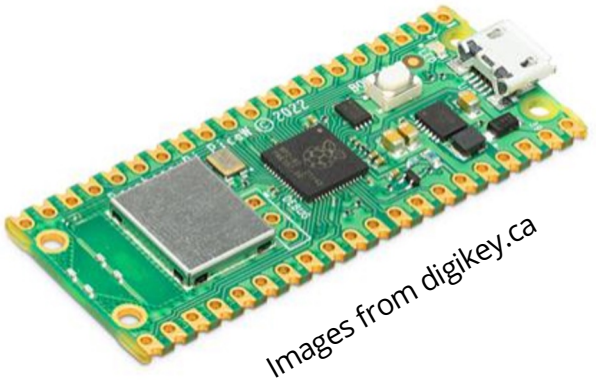
github.com/StevenKnudsen/HackED2025Workshop

Raspberry Pi Pico and Pico W

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

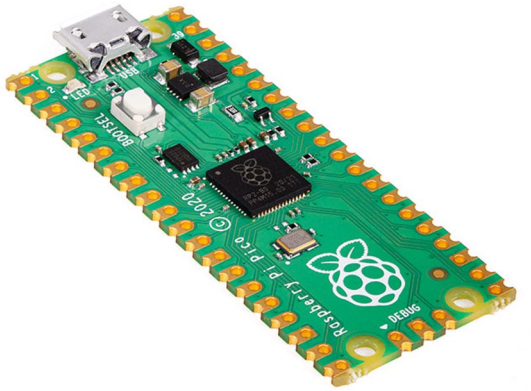
- RP2040 microcontroller chip designed by Raspberry Pi in the United Kingdom
- Dual-core Arm Cortex M0+ processor, flexible clock running up to 133 MHz
- 264 kB of SRAM, and 2MB of on-board flash memory
- USB 1.1 with device and host support
- Low-power sleep and dormant modes
- Drag-and-drop programming 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

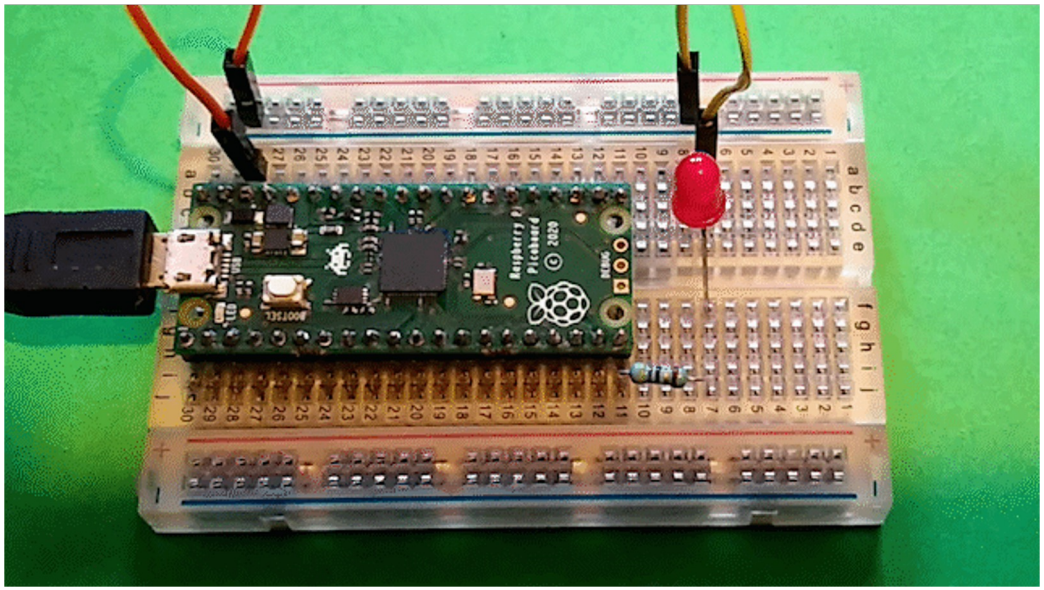
$CA8.34 at canada.newark.com

Images from digikey.ca

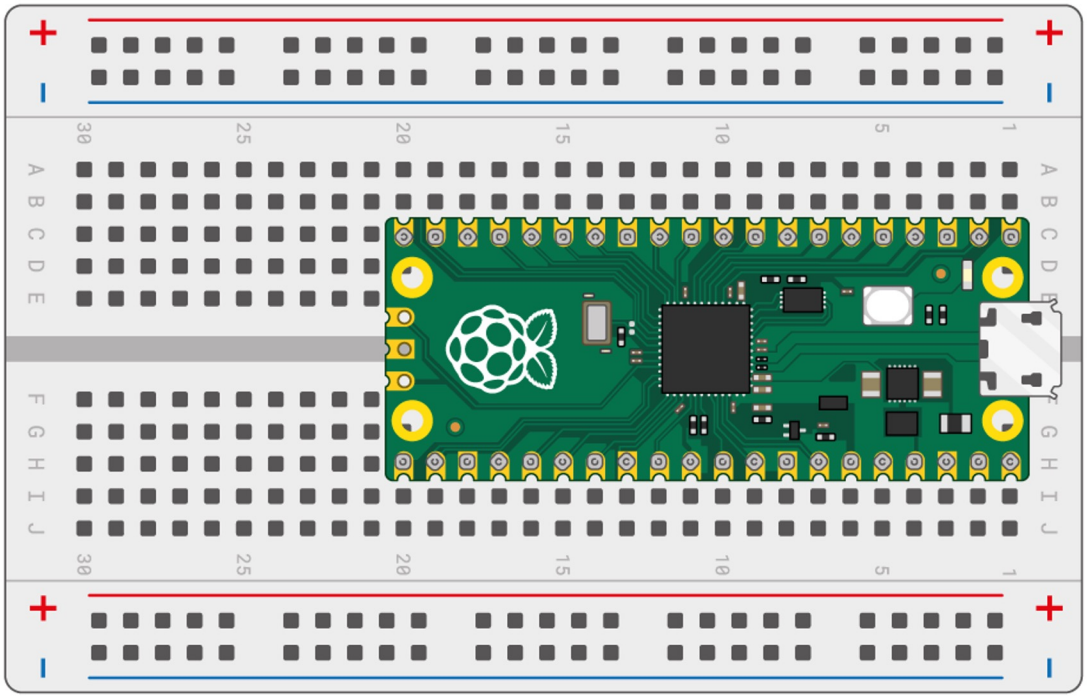
Getting started with the Pico - MicroPython

The main reference is a very good tutorial

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

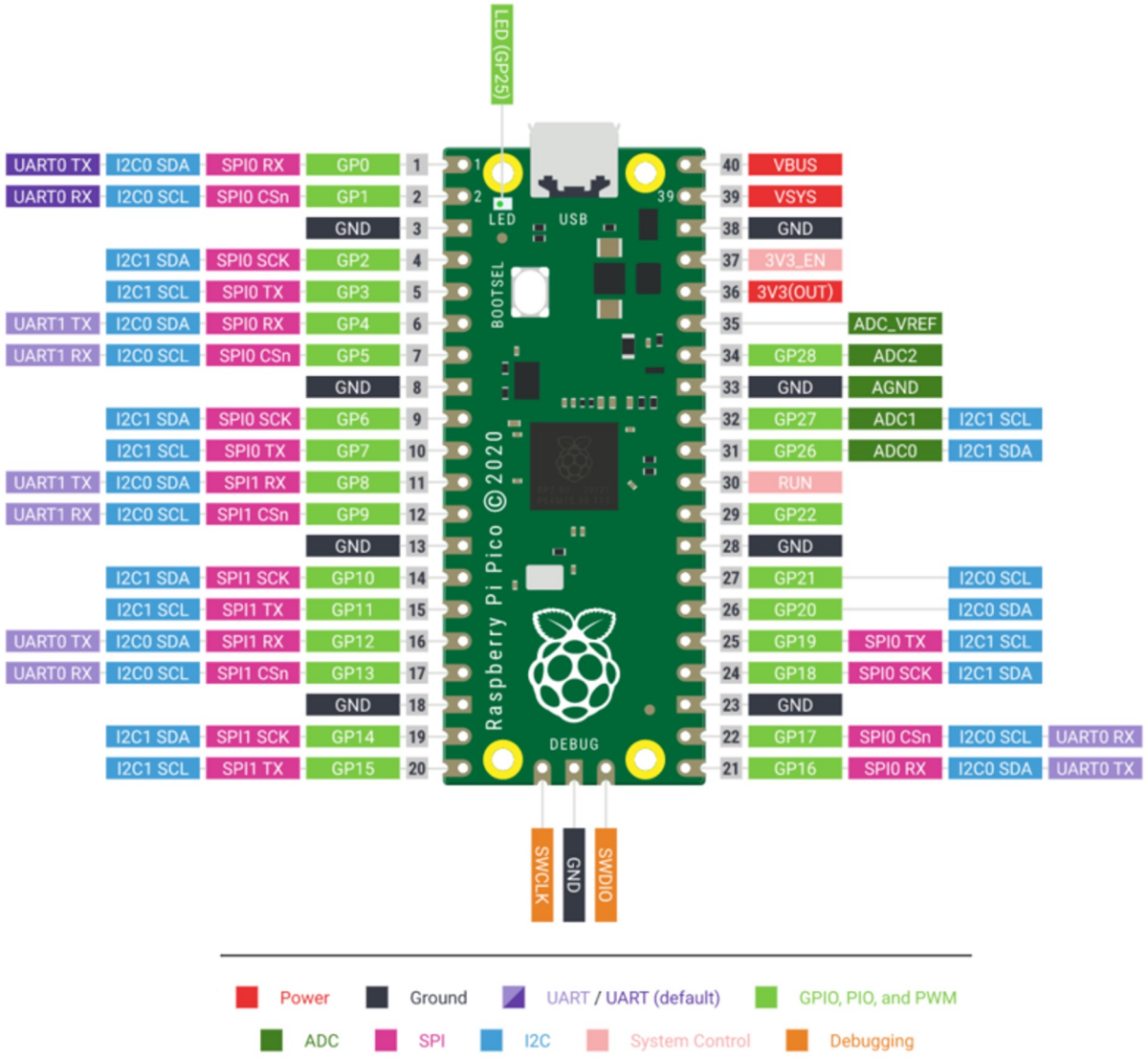


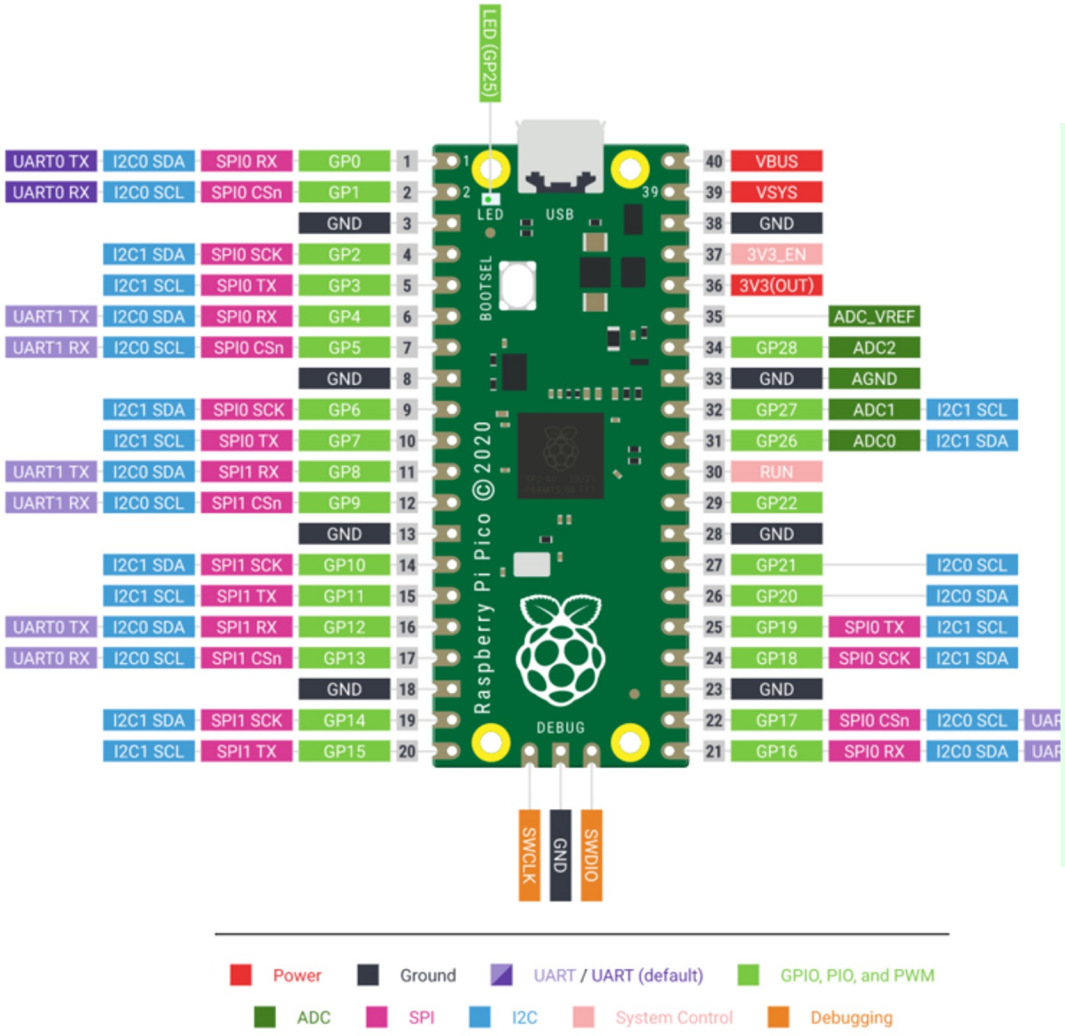
Pico on breadboard

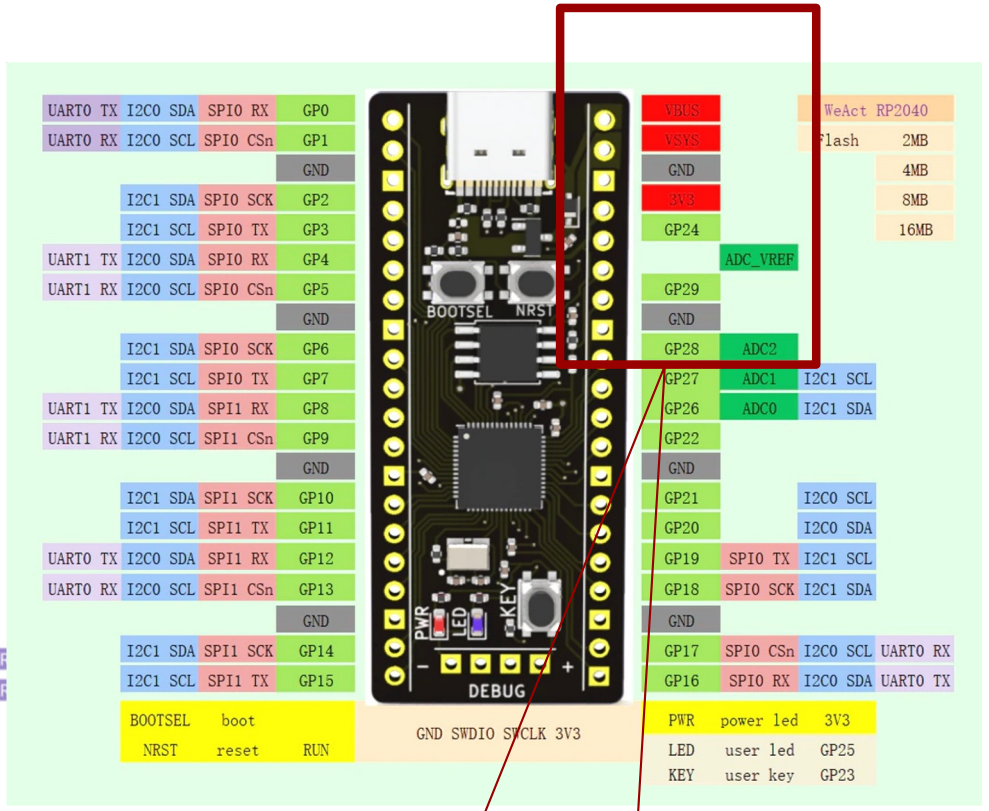


https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/1

RPi Pico Pins




LED(GP25)
LED
USB
BOOTSEL
Raspberry Pi Pico © 2020
DEBUG
SWCLK
GND
SWDIO
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
21 GP16 SPI0 RX I2C0 SDA
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
BOOTSEL boot
NRST reset RUN
BOOTSEL
NRST
PWR
LED
KEY
DEBUG
GND SWDIO SWCLK 3V3
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
PWR power led 3V3
LED user led GP25
KEY user key GP23
WeAct RP2040
Flash 2MB
4MB
8MB
16MB

NOT THE SAME!

Assumptions

- You know how to program in at least one of the languages
  - Python or C
  - Rust, preferably...

- You have Pico
  - You have the official Pico debug tools, or
  - You have a Segger debug probe, or
  - A second Pico

Installing Rust

- Visit rust-lang.org/tools/install to install `rustup`
- Tool that installs and manages Rust
- Rust is developed on a 6-week sprint, good to update often

```
rustup self update
rustup update stable
```

rp-rs

- Github project that supports a Rust HAL (hardware abstraction layer) that enabled embedded programming in Rust on the Raspberry Pi Pico family

```
rustup target add thumbv6m-none-eabi
cargo install flip-link
```

May have dependency issues. Ubuntu 24.04 needed libudev-dev installed, for example

VSCode

- Install `rust-analyzer, Debugger for probe-rs`
- Install `crates` — Keeps crates up to date

- Get `debugprobe_on_pico.u2f` from github.com/raspberrypi/debugprobe/releases
    - Drag and drop programming with pico in bootloader mode

Pico in boot select (bootsel) mode

Unplug Pico from USB

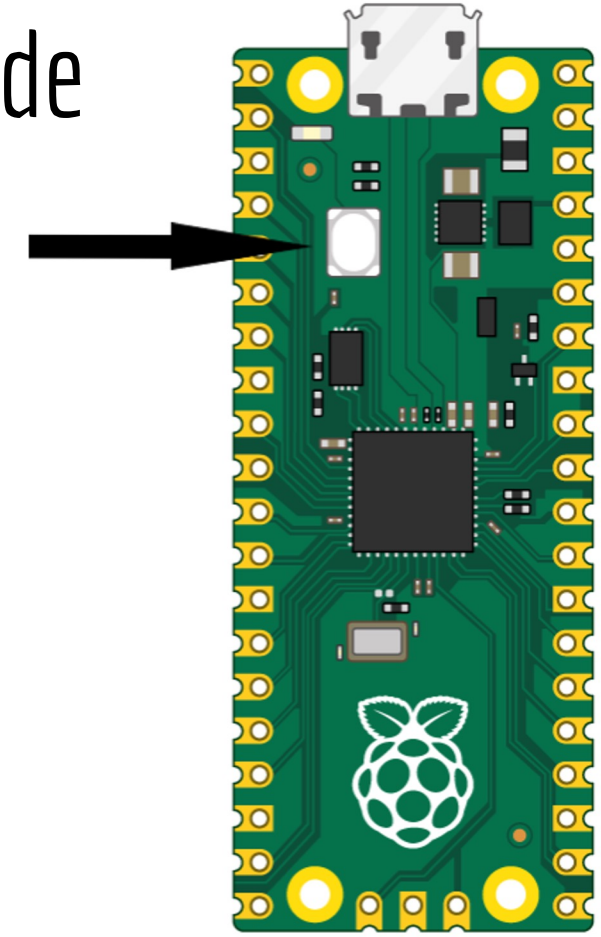
Press and hold bootsel button

Plug Pico into USB and wait a few seconds

Release bootsel button

Pico appears as storage device

Drag and drop u2f image



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

Probe-rs - gdb effectively

- Use probe-rs
- Check for pico probe and show output
- Could use `cargo install --locked probe-rs-tools`, but instead

```
$ curl --proto '=https' --tlsv1.2 -LsSf https://github.com/probe-rs/probe-
rs/releases/latest/download/probe-rs-tools-installer.sh | sh
```

```
$ probe-rs list
The following debug probes were found:
[0]: Debugprobe on Pico (CMSIS-DAP) – 2e8a:000c:E6616407E33CA42F (CMSIS-DAP)
```

- Be sure to set up `udev` rules in Linux

Blinky program example

- Borrow from Murray Todd on Medium

  https://murraytodd.medium.com/learning-rust-with-embedded-programming-on-rp2040-e784389d2d3d

- Use hardware abstraction layer (HAL) from rp-rs project
- https://github.com/rp-rs
- Go to rp2040-project-template, clone

```
git clone https://github.com/rp-rs/rp2040-project-template.git
```

VSCode work

- Open the template folder in VSCode
- Check in .cargo/config.toml that

```
runner = "probe-rs run --chip RP2040 --protocol swd”
```

- In Cargo.toml rename the project to `rp2040-blinky`
- In .vscode/launch.json change

  `"programBinary": "target/thumbv6m-none-eabi/debug/rp2040-project-template”`

  to

  `"programBinary": "target/thumbv6m-none-eabi/debug/rp2040-blinky"`

VSCode work

- In .vscode/launch.json, uncomment the line `"svdFile": "./.vscode/rp2040.svd"`
- Manually download rp2040.svd from

  `https://raw.githubusercontent.com/raspberrypi/pico-sdk/1.3.1/src/rp2040/hardware_regs/rp2040.svd`

  and place it in the .vscode directory so that the debugger can find it

VSCode work

- In Cargo.toml, uncomment the two lines with `rp2040*`

```
# If you're not going to use a Board Support Package you'll need these:
rp2040-hal = { version="0.10", features=["rt", "critical-section-impl"] }
rp2040-boot2 = "0.3”
```

  so that the HAL is included

- Time to run the debugger!

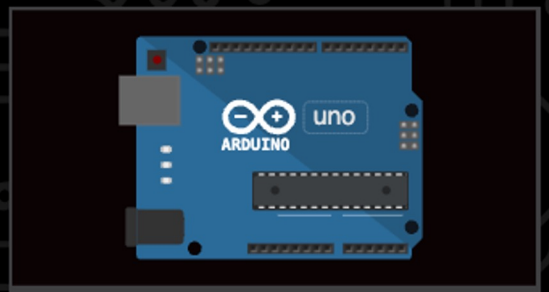
One more tool – Wokwi.com

Wokwi

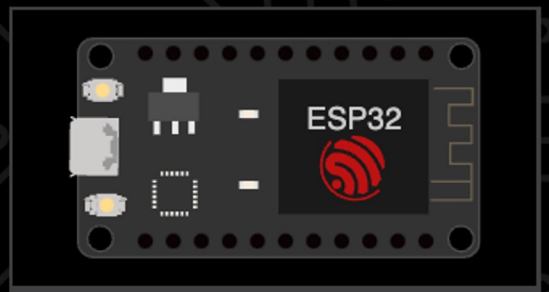
Simulate IoT Projects in Your Browser

Discord Community LinkedIn Group

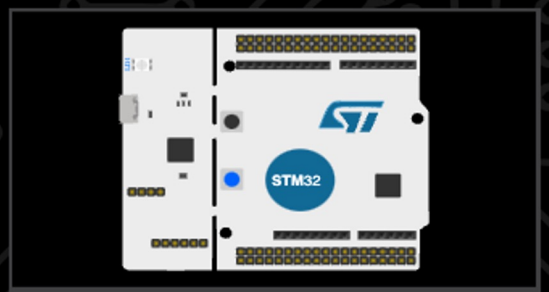
Simulate with Wokwi Online



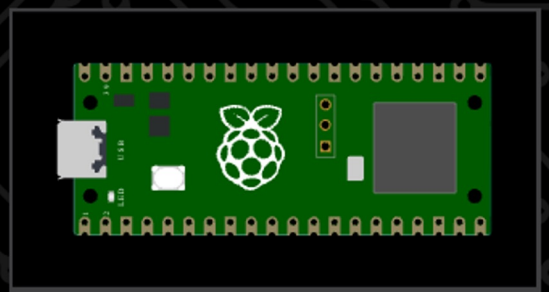
Arduino (Uno, Mega, Nano)



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.

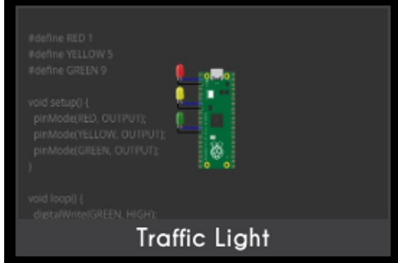
Here are some quick examples of things you can make with Wokwi:

- 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

Raspberry Pi Pico Simulator

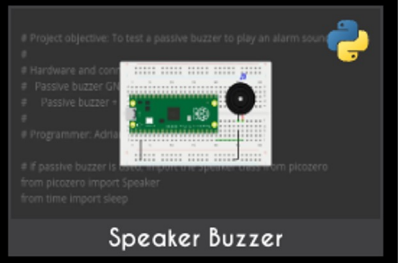
A faster way to prototype Pi Pico projects

Featured projects

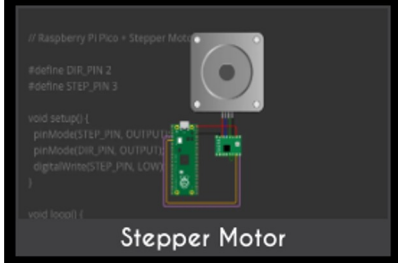


Traffic Light

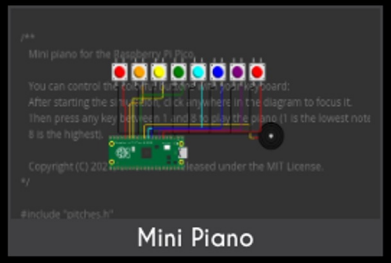
LCD1602



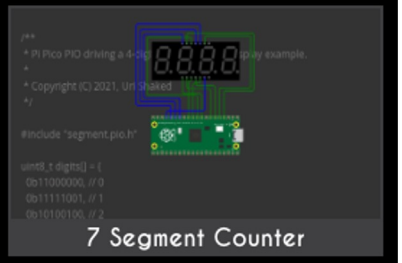
Speaker Buzzer



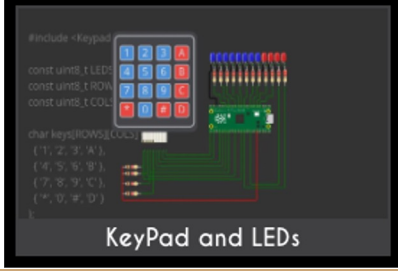
Stepper Motor



Mini Piano

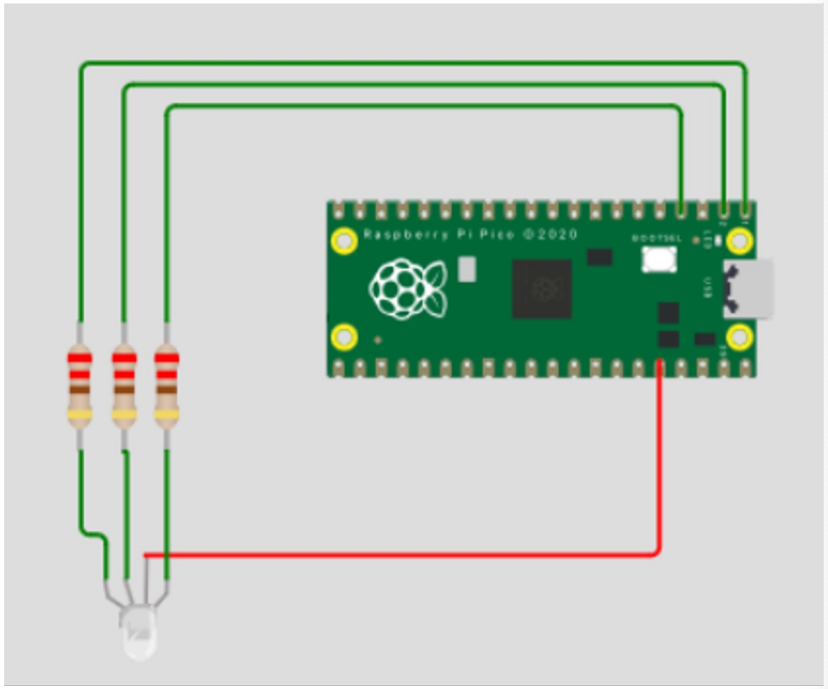


7 Segment Counter



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)
23
```

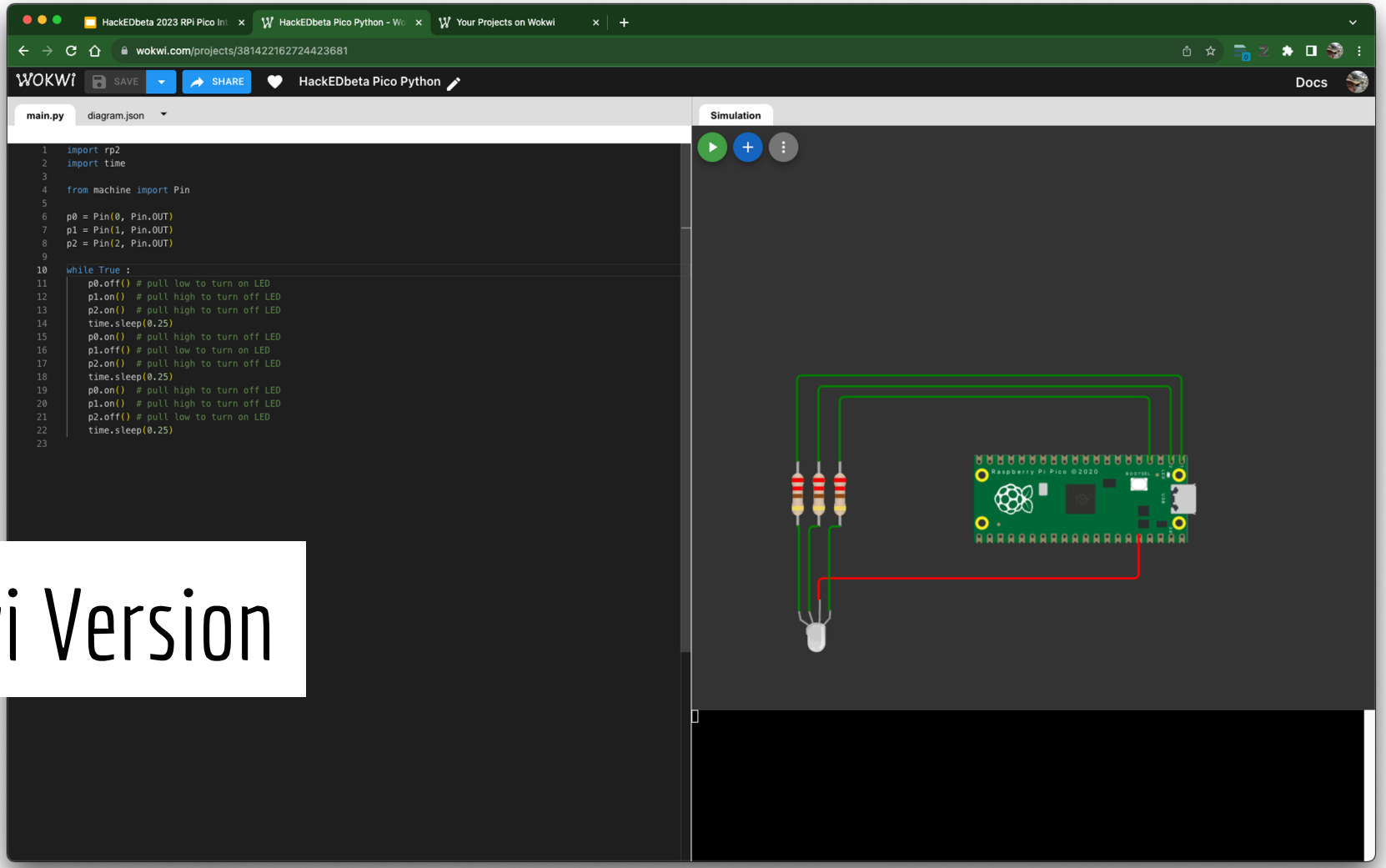
Shell

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

  MPY: soft reboot
```

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

Wokwi Version



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

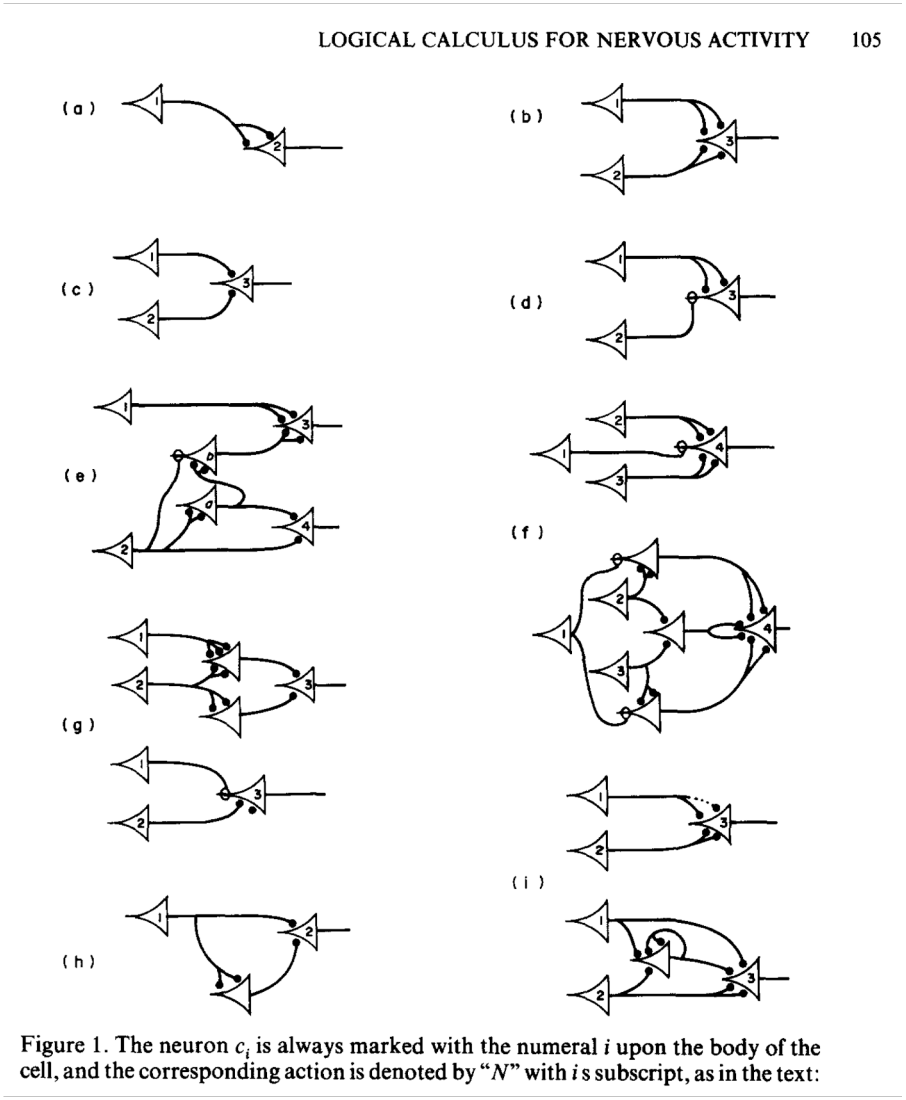


Figure 1. The neuron c_i is always marked with the numeral i upon the body of the cell, and the corresponding action is denoted by "N" with i s subscript, as in the text:

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”

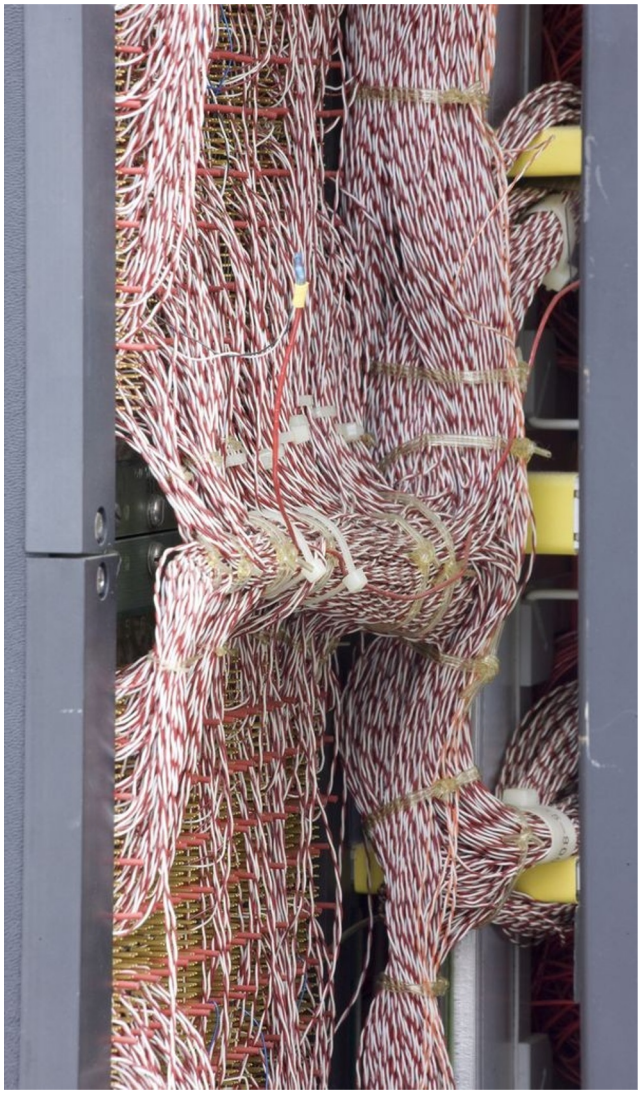
Keywords : Clones; Plug compat; IBM

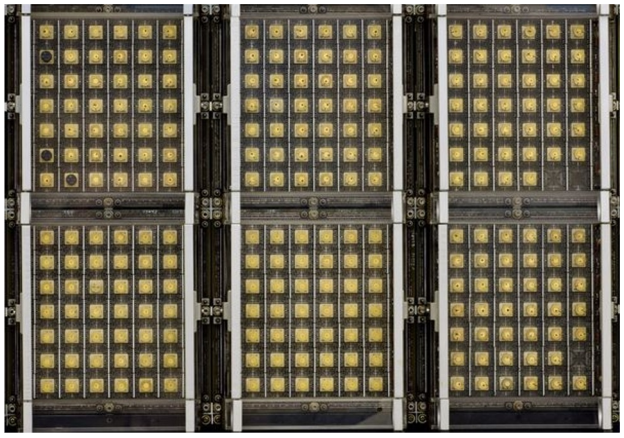
Speed : 3.5 MIPS

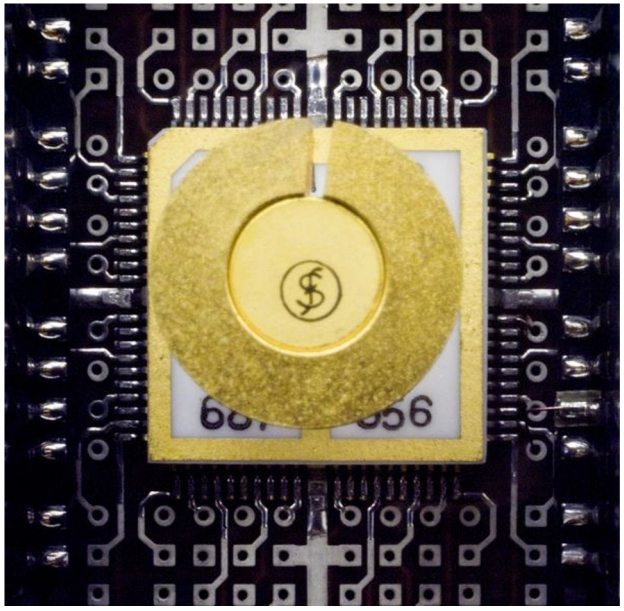
Memory Size : up to 8MB

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%

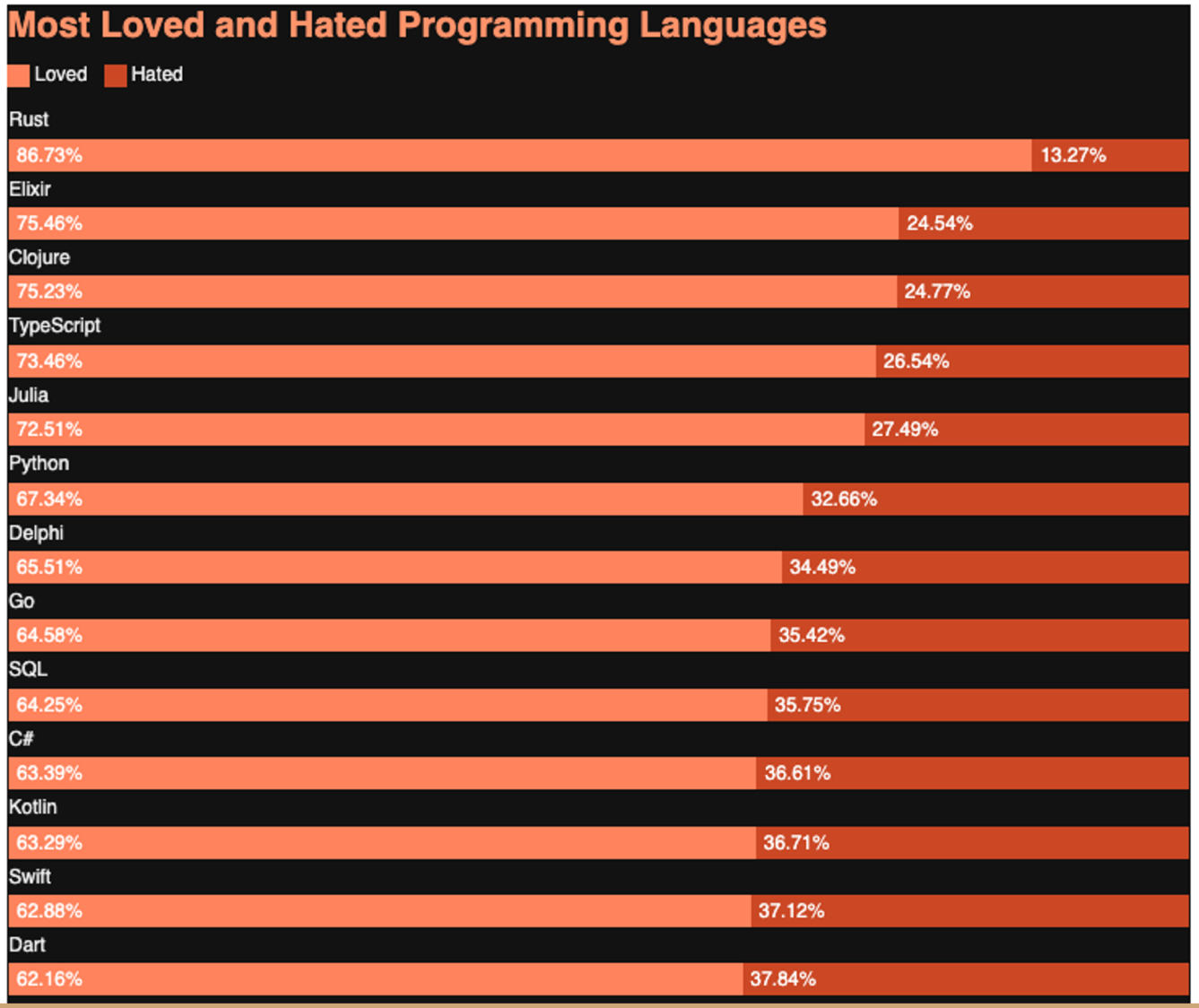
Elixir 75.46%

Clojure 75.23%

TypeScript 73.46%

Julia 72.51%

Python 67.34%

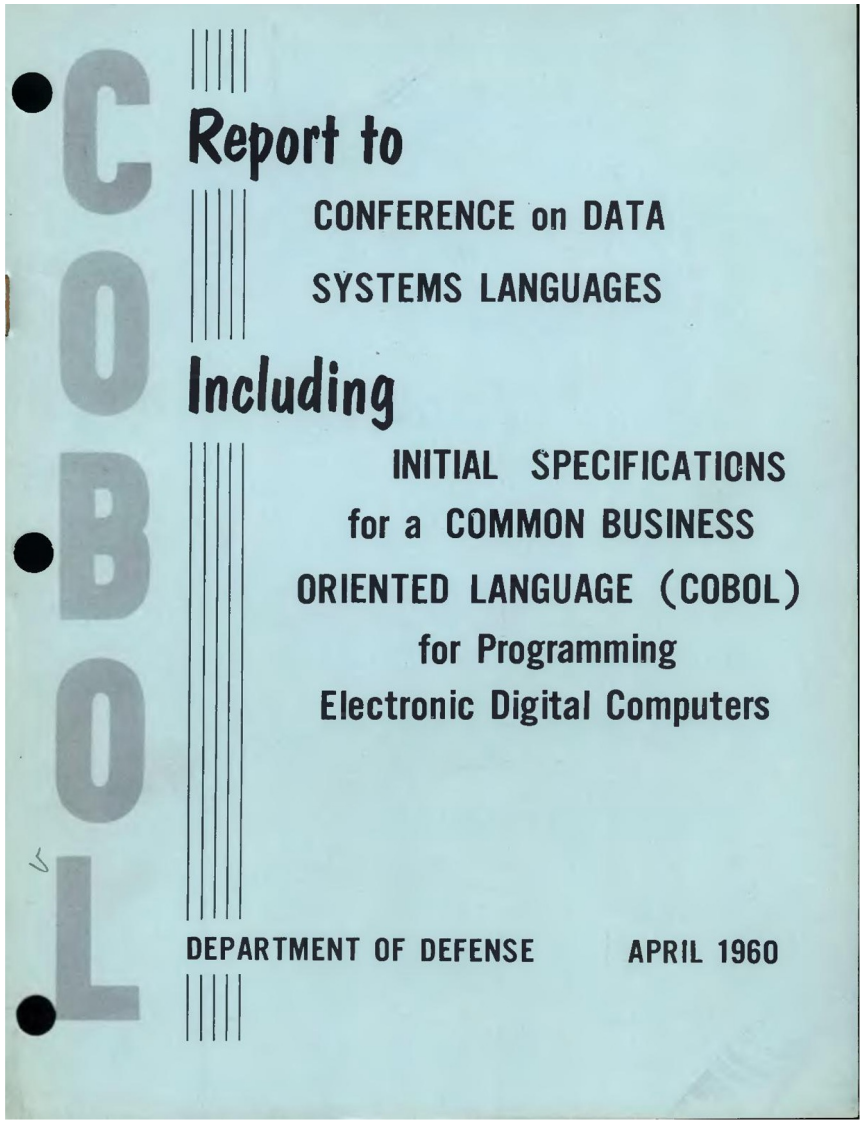


Most hated

VBA (78.56 percent).

COBOL (hated by 79.96 percent of developers)

MATLAB (80.84 percent), and

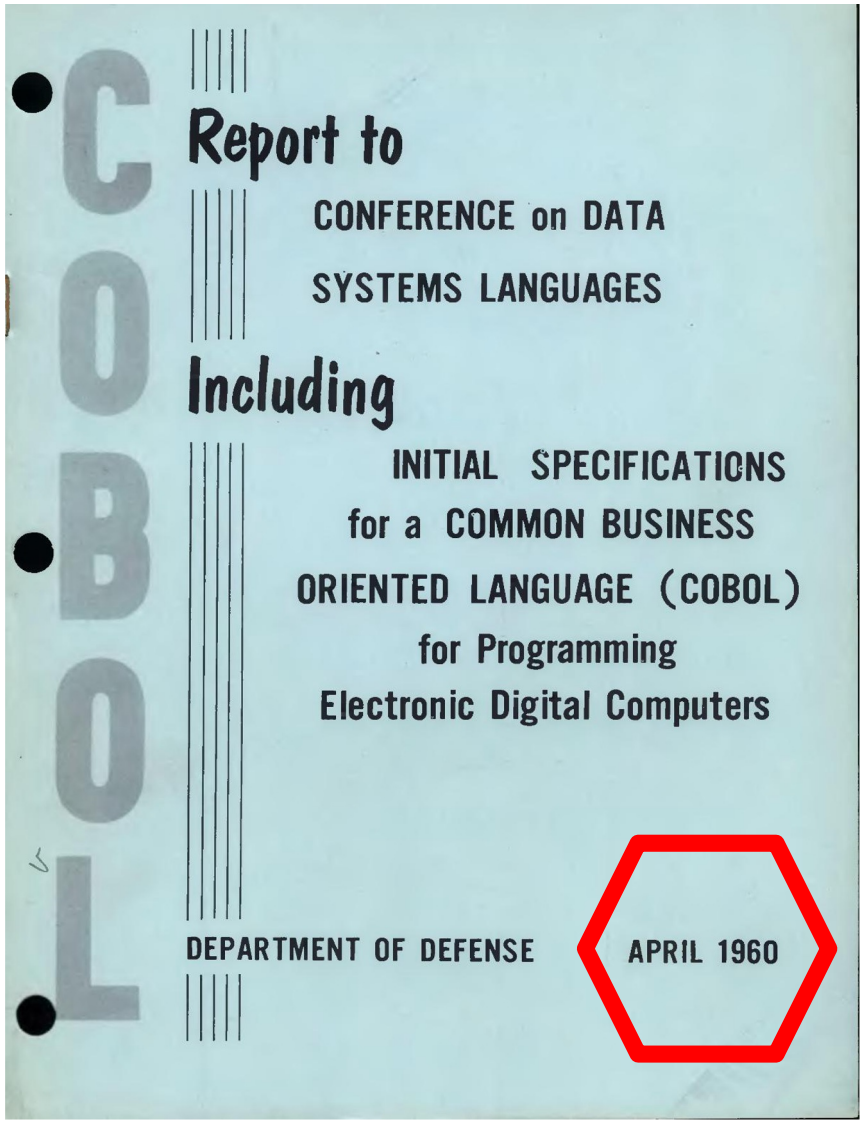


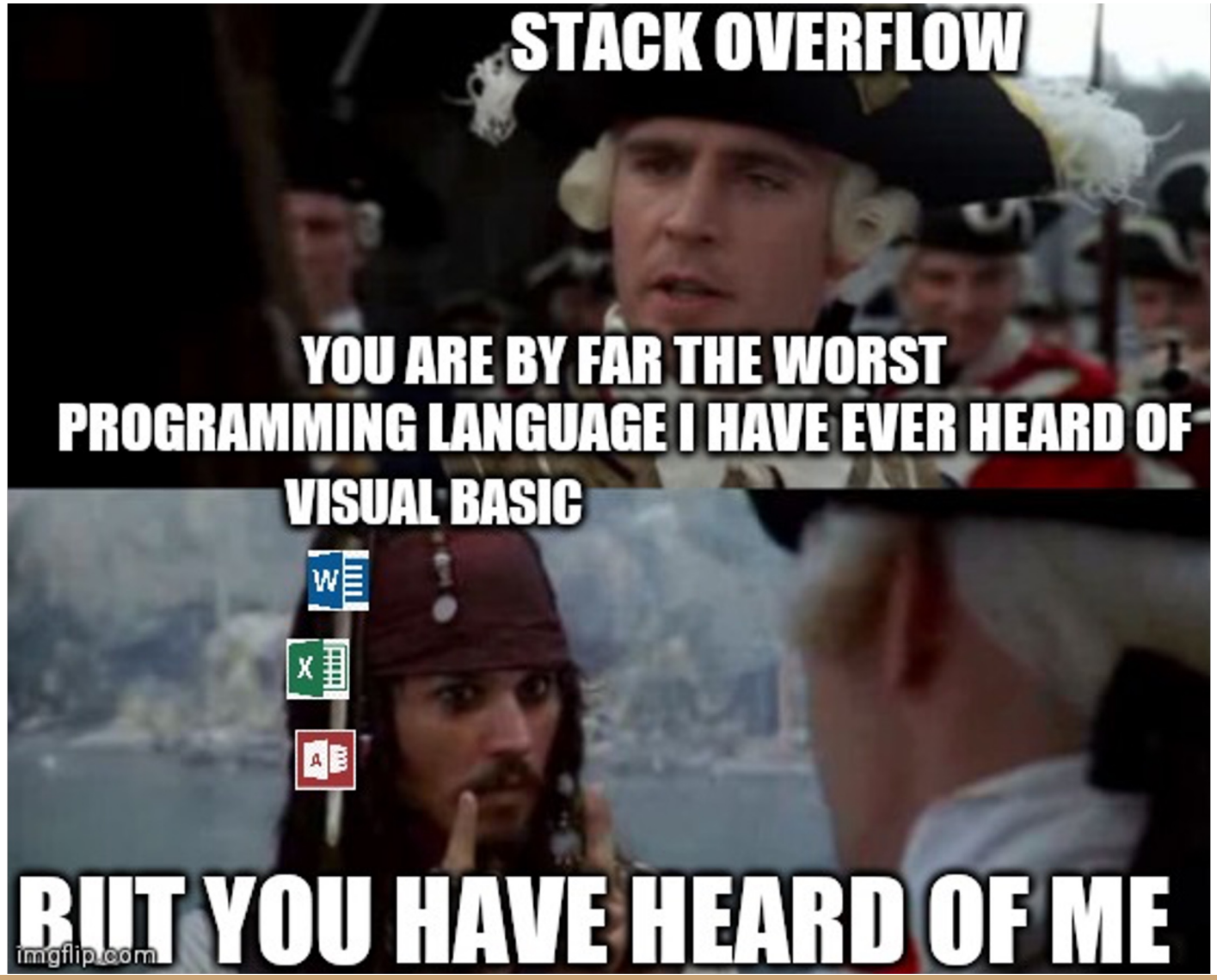
Most hated

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
imgflip.com