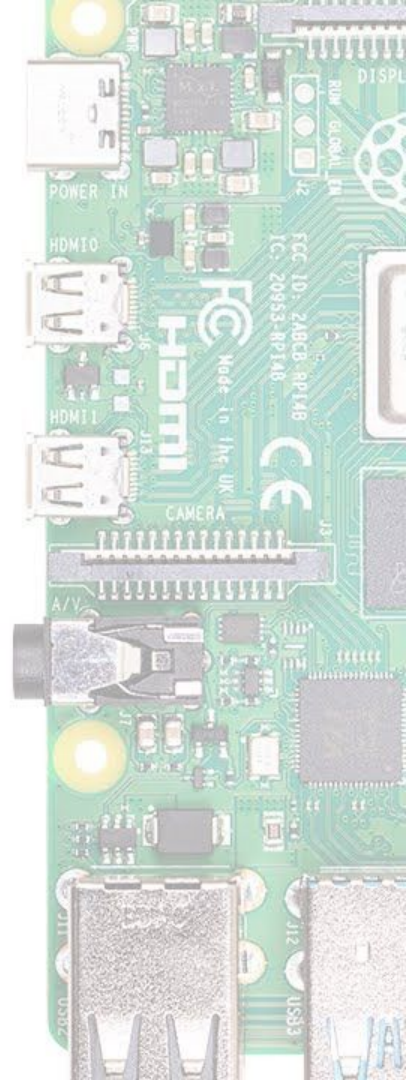


Monitoria

Microcontroladores 2021.2

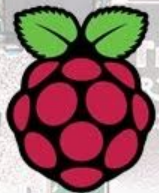
Bruno Gabriel F. Sampaio



Introdução Raspberry Pi Pico



- Uma breve introdução: Família Raspberry Pi e projetos
- O lançamento do Raspberry pi Pico
- Explorando ideias com o Pico

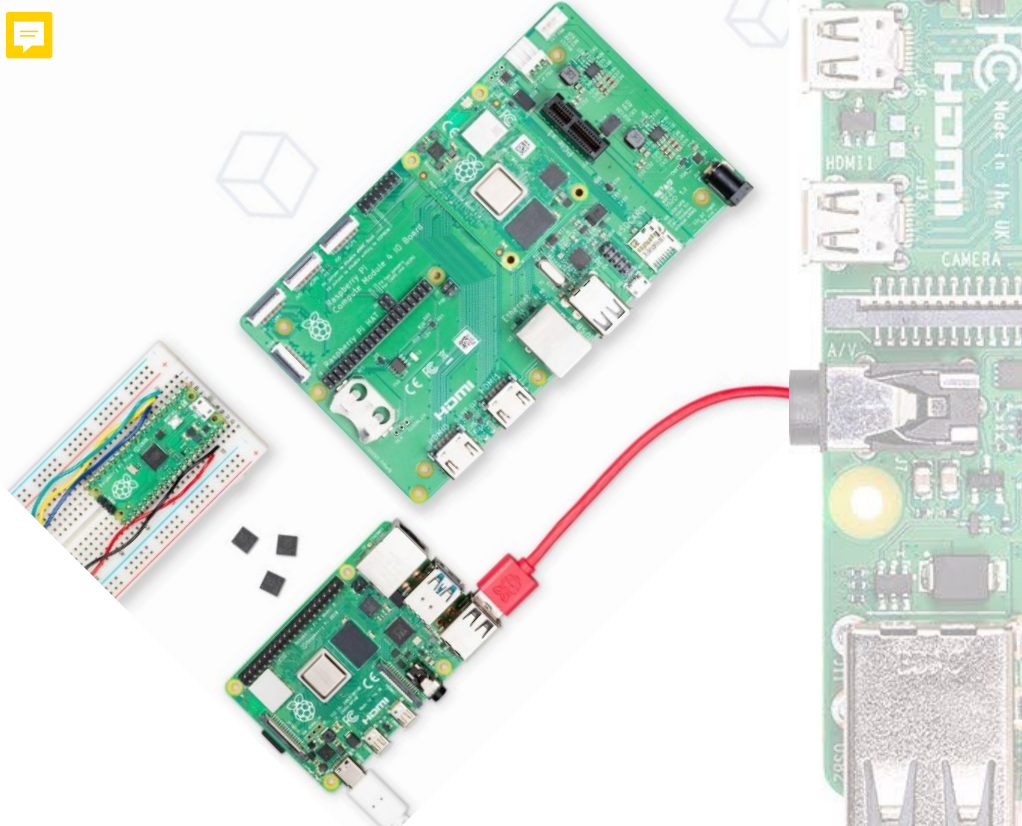


Uma breve introdução acerca da família Raspberry Pi

<https://www.raspberrypi.org/>

A Raspberry Pi Foundation é uma instituição sem fins lucrativos com sede no Reino Unido que trabalha para colocar o poder da computação e da produção digital nas mãos de pessoas em todo o mundo.

“Fazemos isso para que mais pessoas possam aproveitar o poder da computação e das tecnologias digitais para trabalhar, para resolver problemas que são importantes para elas e para se expressarem de forma criativa.”
-RaspberryOrganization



Microprocessador Raspberry Pi - Modelo B

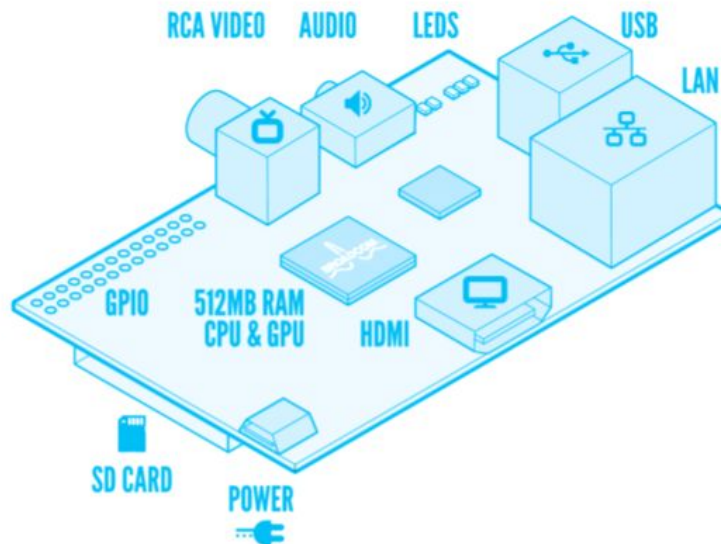
<https://botlanta.org/AHRC/uploads/pdf/RaspberryPi.pdf>

É considerado um mini computador, sendo uma solução barata para ser usada no aprendizado de conceitos de computação e/ou projetos de automação.

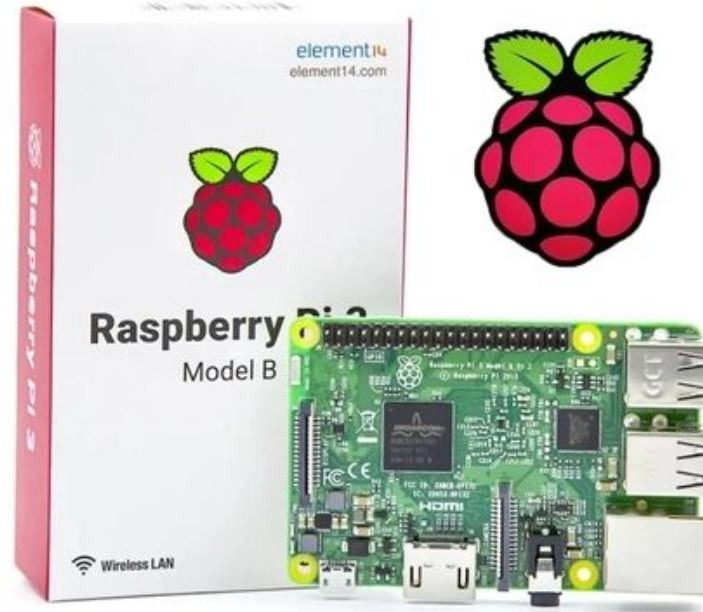
É um sistema generalista, podendo ser usado com muitos propósitos.

RASPBERRY PI 3 MODEL B

Processador	Processador Broadcom BCM2837 64 bits Quad-core
Clock	1.2GHz
Memória	1 GB SDRAM @400MHz
Portas USB 2.0	4
Conectividade	Ethernet 10/100, Wifi 802.11n e bluetooth 4.0
GPIO	40 pinos
Slot cartão	micro SD
Alimentação	5V via conector USB ou GPIO

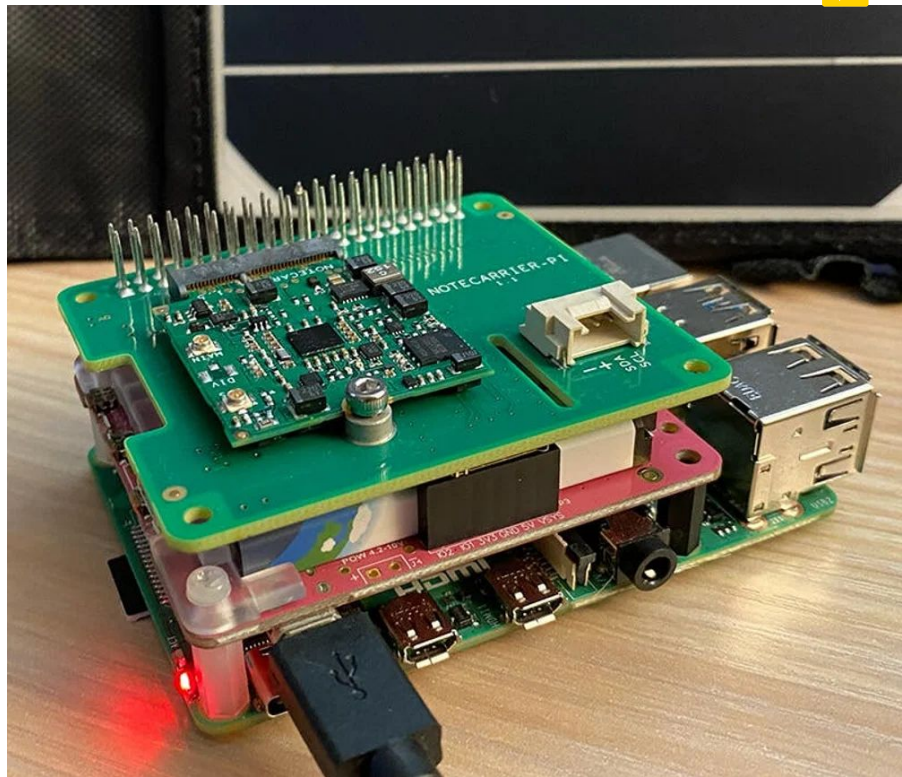


Projetos usando Raspberry pi



De acordo com: <https://all3dp.com/pt/1/melhor-projeto-raspberry-pi/>

Minerador de Bitcoin acionado por energia solar



A mineração de Bitcoins consome 110 terawatts por hora ao ano, o que equivale a 0,55% da produção mundial de eletricidade: é uma quantidade enorme de energia!

Usando um Raspberry Pi com uma PiJuice*, você pode criar uma “equipe de mineração” de criptomoedas alimentado por energia solar com informes baseados na nuvem e em um sistema autônomo. Nada que irá torná-lo rico, já que a potência de cálculo de um Raspberry Pi não está à altura das atuais máquinas de mineração de criptomoedas, mas ao menos você poderá ir dormir com a consciência tranquila.

Raspberry pi 4

*PiJuice: Shield de bateria para alimentação do Raspberry pi

Dispensador de cereais com inteligência artificial

Este dispositivo baseado em Raspberry Pi é nada menos do que um dispensador de alimentos movido a inteligência artificial. Você só coloca a sua tigela debaixo do cereal de que mais gosta. O algoritmo de inteligência artificial então detecta o seu rosto e dispensa os cereais adequados abrindo uma tampa.

Raspberry pi 3



Máquina de fliperama



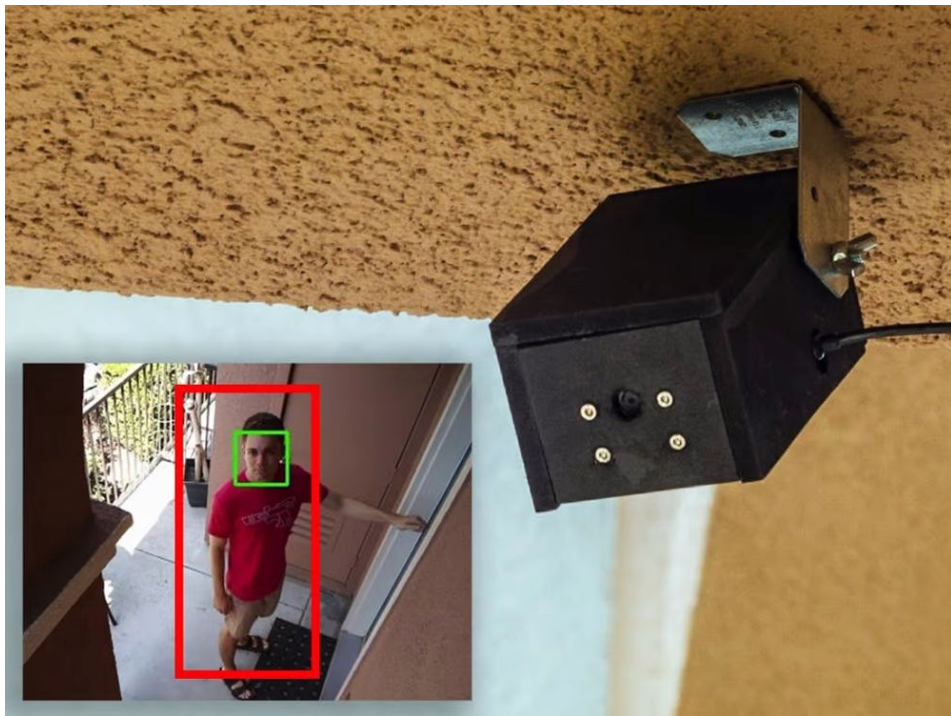
Indo muito além da típica instalação de jogos clássicos em Raspberry Pi, esta bela cabine de fliperama bartop usa joysticks e botões autênticos para dois jogadores. Este projeto requer alguma habilidade com carpintaria e um pouco de experiência com fiação.

Raspberry pi 4

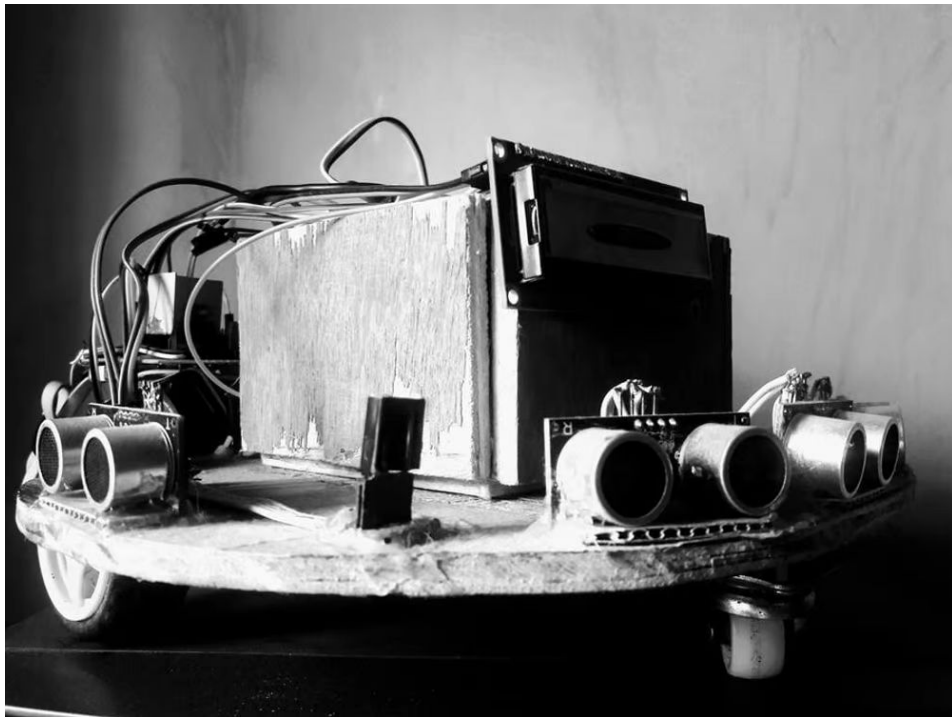
Dispensador de cereais com inteligência artificial

Que utilidade tem uma câmera de vigilância se você tem que monitorá-la constantemente? Esta Câmera Inteligente de Segurança facilita bastante as coisas ao detectar mudanças e enviar notificações a você. Atualmente, o Google pode encaminhar a você um e-mail com a imagem de qualquer objeto que tenha sido detectado.

Raspberry pi Zero



Robô de limpeza autônomo



Robôs... o que seria da humanidade sem eles? Neste projeto Raspberry Pi você irá construir um robô de limpeza que aspira os pavimentos. Mas vai além disso: Ele acorda você pela manhã, lembra da hora de tomar algum remédio e ainda pode ser usado como brinquedo controlado remotamente por crianças. Ele também retorna à sua estação de carga quando a bateria está baixa, e supervisiona a atividade no interior da sua casa quando você estiver fora.

Raspberry pi 4

Porque não usar um
Microprocessador ao invés de
um Microcontrolador ?

VS



Propósito

Microprocessador

- Solução mais cara.
- É capaz de rodar um SO.
- Processamento de dados complexos:
 - Visão computacional
 - Mineração de dados
 - Treinamento de redes neurais
- Projetos generalistas.



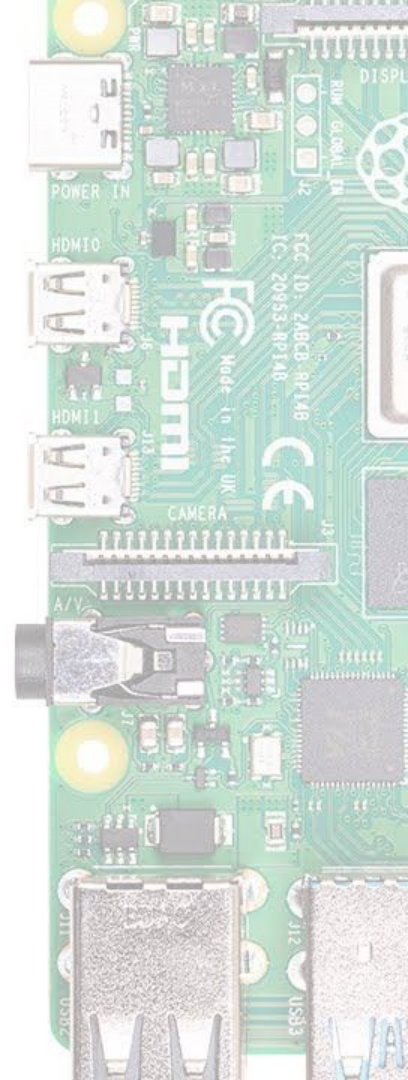
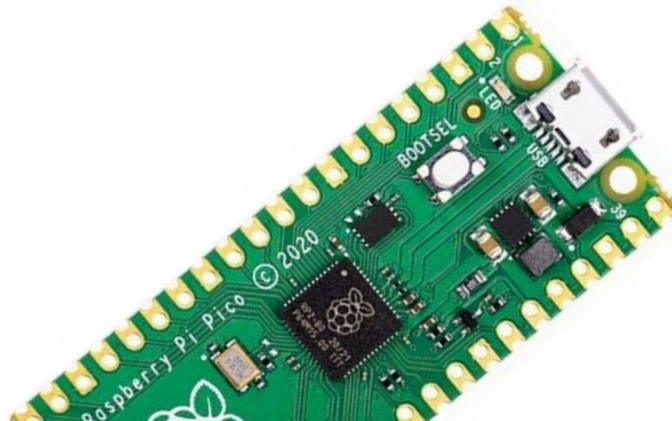
Microcontrolador

- Solução mais barata.
- Não possui SO.
- Processamento de dados simples:
 - Programação a nível lógico
 - Controle de GPIOs
 - Acionamentos
- Projetos especializados.



Criação do Pico

- A Raspberry Pi Foundation anunciou em 2020, a criação de um microcontrolador que custa apenas US\$ 4 (R\$49 no Br).
- O Raspberry Pi Pico permite interagir e dar comandos a outros dispositivos, possibilitando a criação de projetos de hardware.
- Além de ser barata, a placa é pequena e consome pouca energia.
- Possuindo o chip RP2040, de fabricação própria da Raspberry pi.



Raspberry Pi Pico

RP2040 microcontroller chip designed by Raspberry Pi in the UK

Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz

264KB on-chip SRAM

2MB on-board QSPI Flash

26 multifunction GPIO pins, including 3 analogue inputs

2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels

1 × USB 1.1 controller and PHY, with host and device support

8 × Programmable I/O (PIO) state machines for custom peripheral support

Supported input power 1.8–5.5V DC

Operating temperature -20°C to +85°C

Castellated module allows soldering direct to carrier boards

Drag-and-drop programming using mass storage over USB

Low-power sleep and dormant modes

Accurate on-chip clock

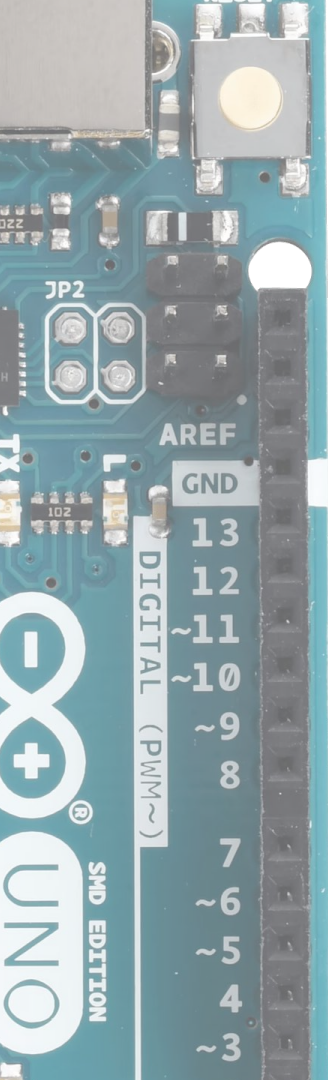
Temperature sensor

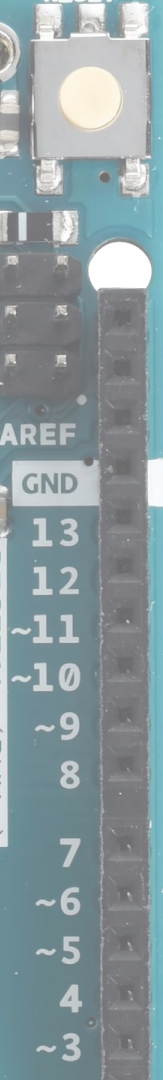
Accelerated integer and floating-point libraries on-chip



Arduino UNO x Pico

```
void setup() {  
    pinMode(LED_BUILTIN, OUTPUT);  
}  
  
void loop() {  
    digitalWrite(LED_BUILTIN, HIGH);  
    delay(1000);  
    digitalWrite(LED_BUILTIN, LOW);  
    delay(1000);  
}  
  
from machine import Pin, Timer  
  
led = Pin(25, Pin.OUT)  
tim = Timer()  
def tick(timer):  
    global led  
    led.toggle()  
  
tim.init(freq=2.5, mode=Timer.PERIODIC, callback=tick)
```

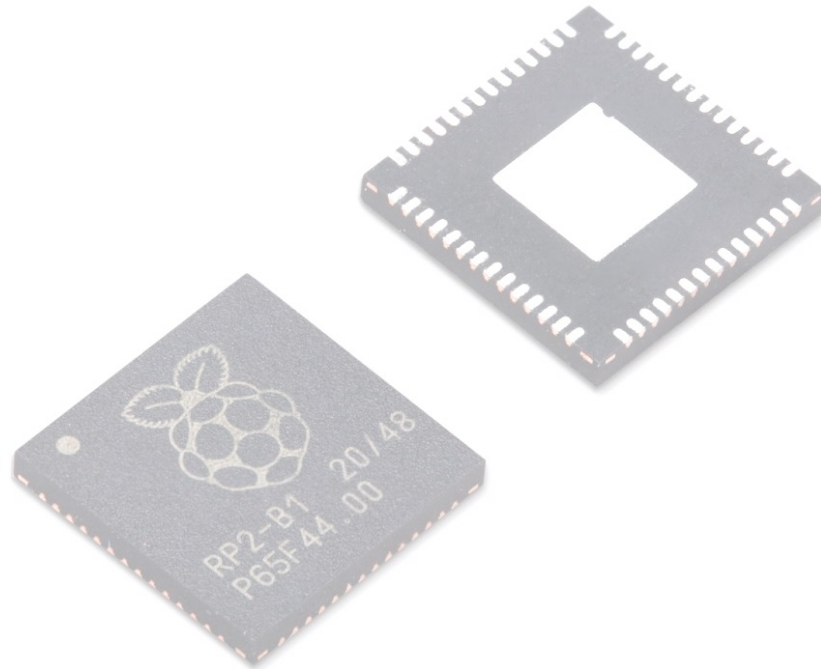




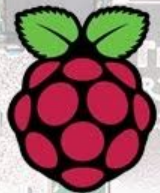
Especificações	Arduino Uno R3	Raspberry Pi Pico
Microcontrolador	ATmega328P	RP2040
Núcleo	Single Core	Dual-core
Arquitetura do núcleo	8 bit RISC	32-bit ARM Cortex M0+
Clock	16 MHz	48 MHz (típico) 133 MHz (<i>overclock</i>)
Memória RAM	2 KB	264 KB
Memória Flash	32 KB	2 MB
EEPROM	1 KB	Não
Linguagem de Programação	Baseado em C	Python e C/C++
Alimentação	5 VDC via USB B	5 VDC via micro USB B
Nível Lógico GPIO	5V	3.3V
GPIO	20 entradas e saídas digitais	26 entradas e saídas digitais
ADC	6 x 10-bit	3 x 12-bit



Hardware



Arquitetura do microcontrolador
RP2040



RP 2 0 4 0

Raspberry Pi

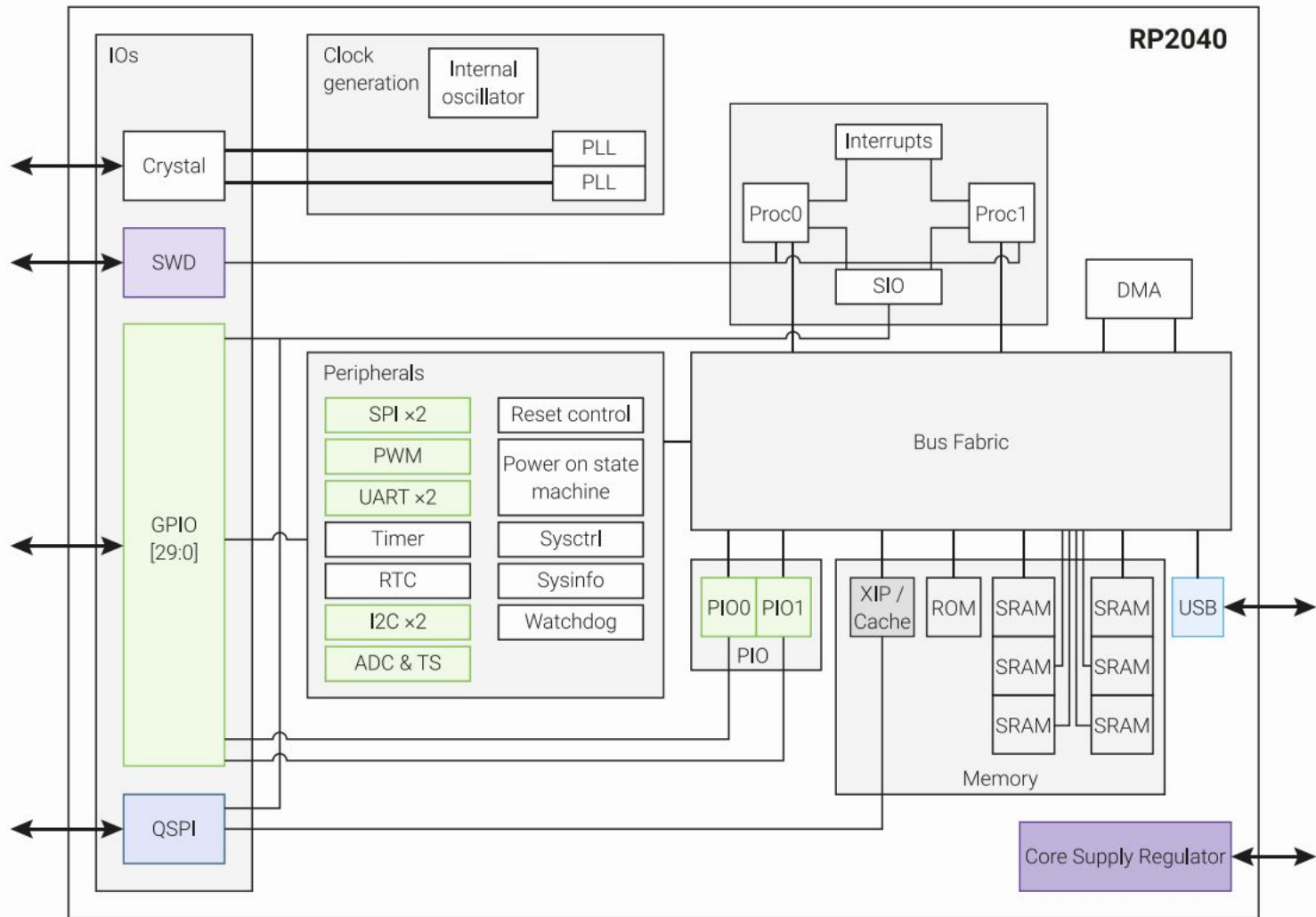
Number of cores

Type of core (e.g. M0+)

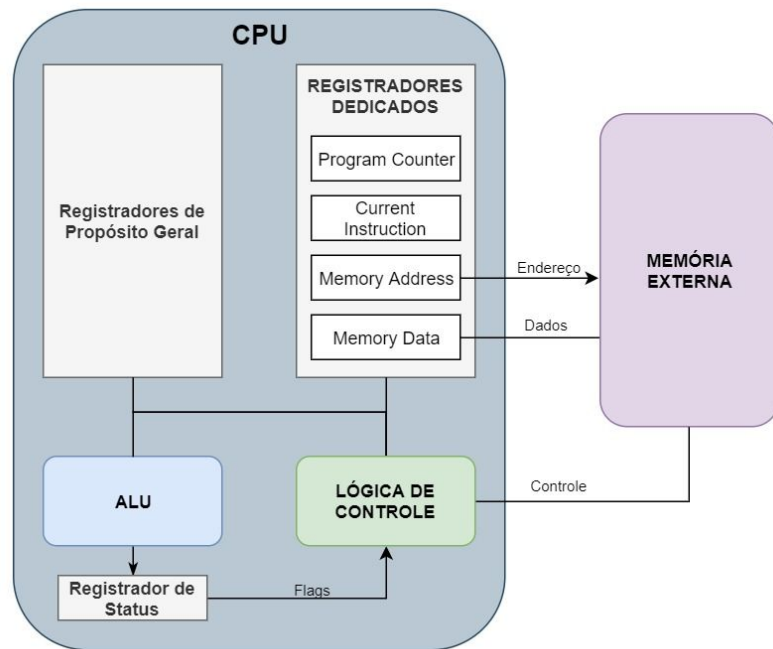
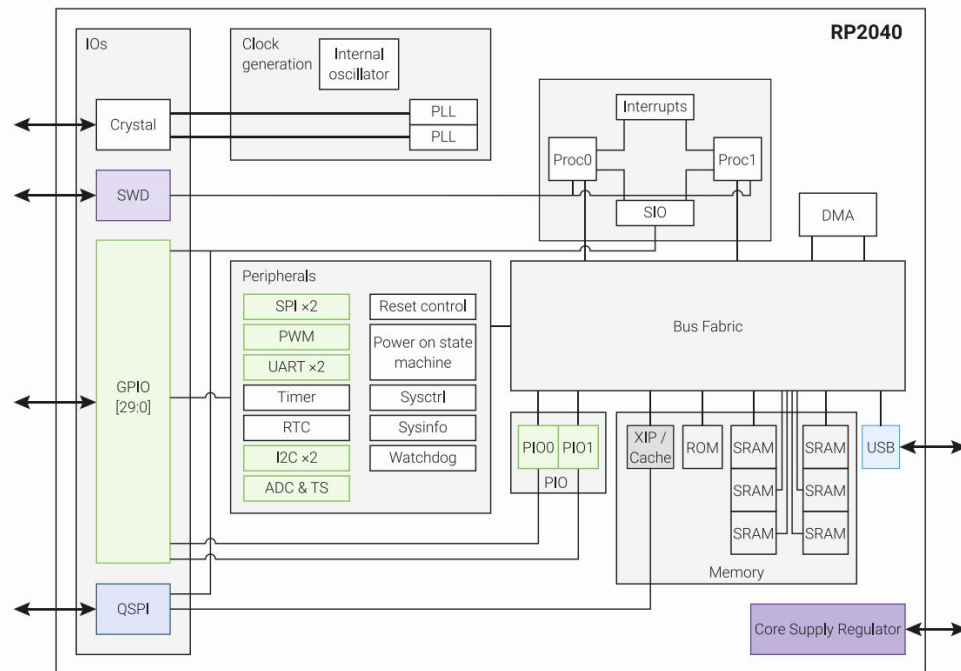
$\text{floor}(\log_2(\text{ram} / 16\text{k}))$

$\text{floor}(\log_2(\text{nonvolatile} / 16\text{k}))$

Arquitectura

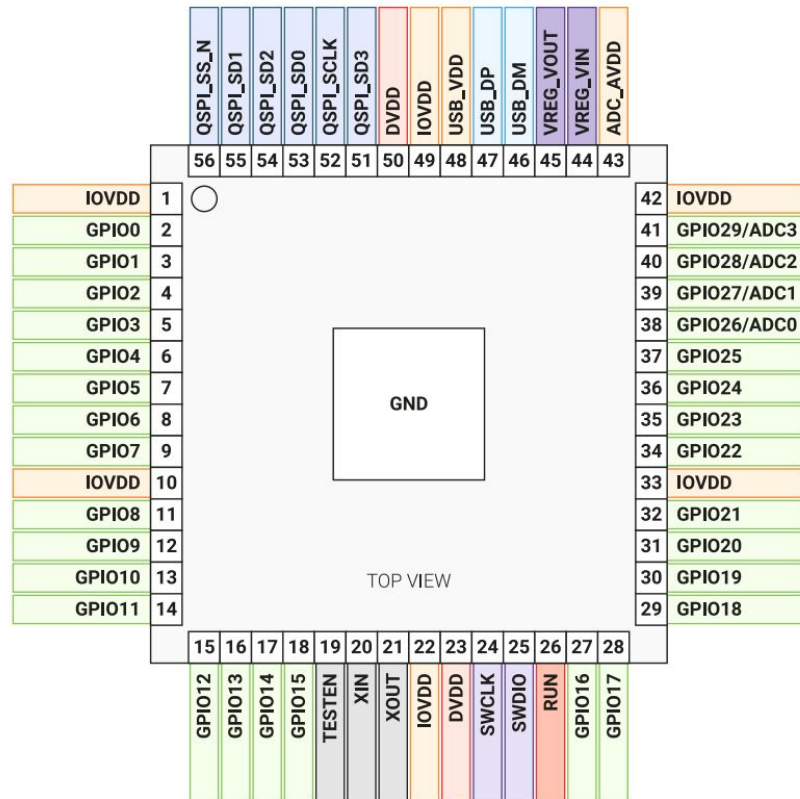


Diferença de arquiteturas Microcontroladores e Microprocessadores



Pinout

RP2040



GPIOx: Entrada e saída digital de uso geral.

GPIOx/ADCy: Entrada e saída digital de uso geral, com função de conversor analógico-digital.

QSPIx: Interface para um dispositivo flash SPI, Dual-SPI ou Quad-SPI, com suporte para execução no local.

USB_DM e USB_DP: Controlador USB, compatível com dispositivo Full Speed e Host Full/Low Speed.

XIN and XOUT: Conexão para o cristal oscilador.

RUN: Pino de redefinição assíncrona global.

SWCLK e SWDIO: Fornece acesso de depuração para ambos processadores.

TESTEN: Pino para teste de fábrica.

GND: Conexão para aterramento externo.

IOVDD: Fonte de alimentação para os pinos GPIO digitais. Tensão nominal 1,8 a 3,3 V.

USB_VDD: Fonte de alimentação para o USB Full Speed PHY interno. Tensão nominal 3,3 V.

ADC_AVDD: Fonte de alimentação para conversor analógico-digital. Tensão nominal 3,3 V.

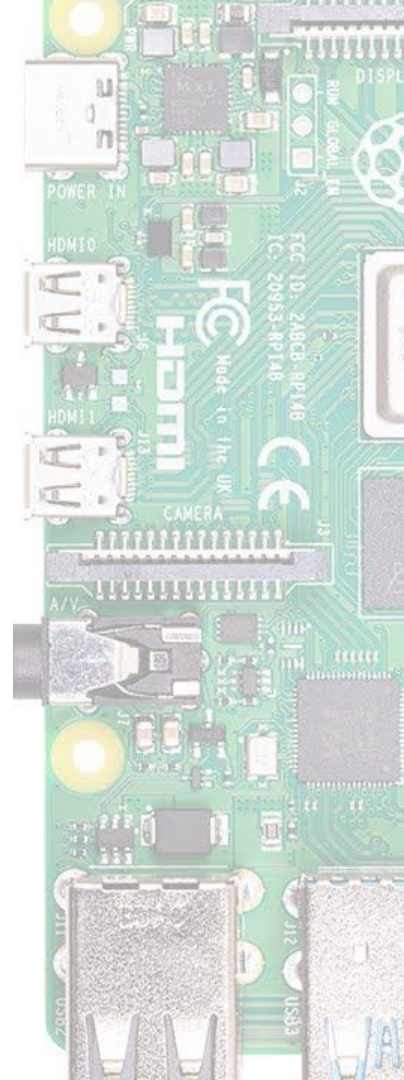
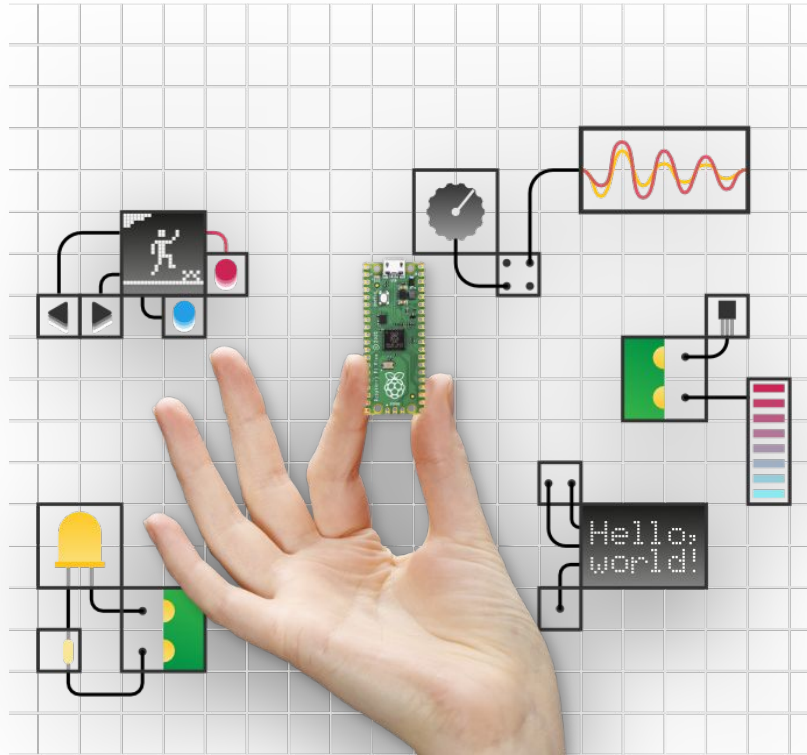
VREG_VIN: Entrada de energia para o regulador de tensão. Tensão nominal 1,8 a 3,3 V.

VREG_VOUT: Saída de energia para o regulador de tensão do núcleo interno. Tensão nominal 1,1 V, corrente máxima de 100 mA.

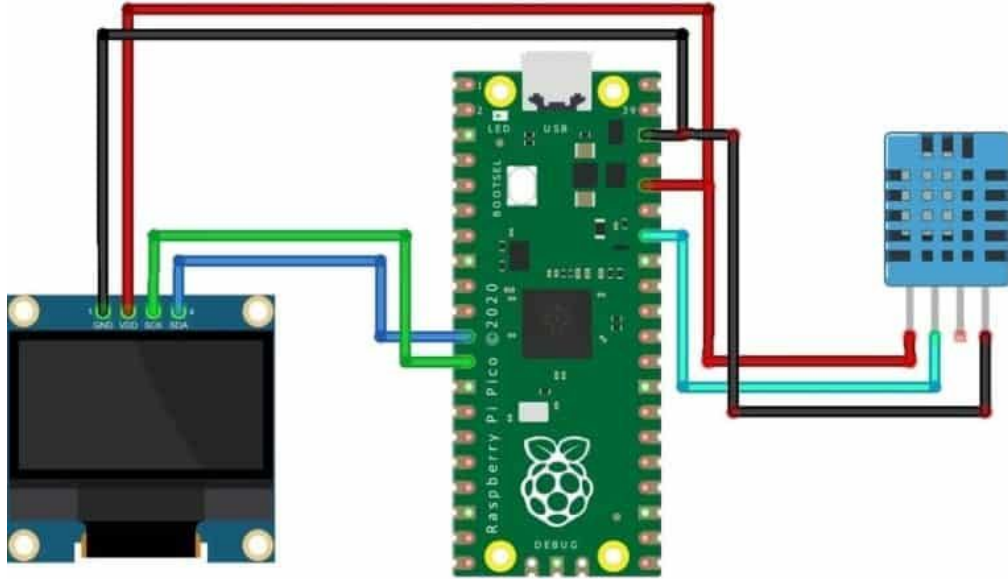
DVDD: Fonte de alimentação digital. Tensão nominal 1,1 V.



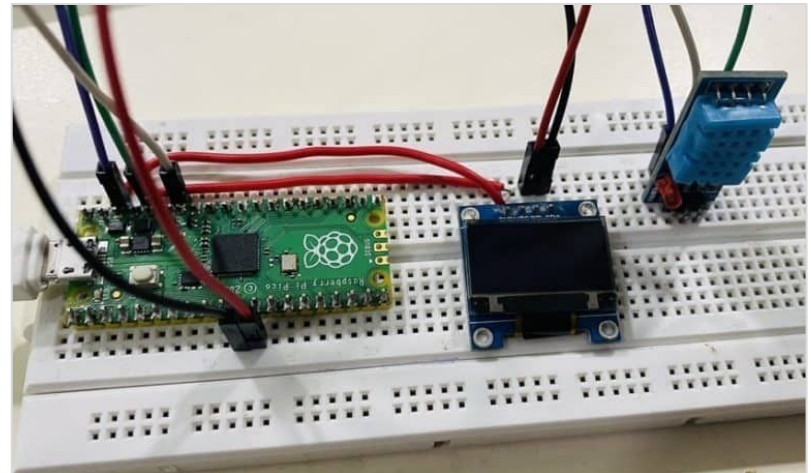
Projetos usando Raspberry Pi Pico



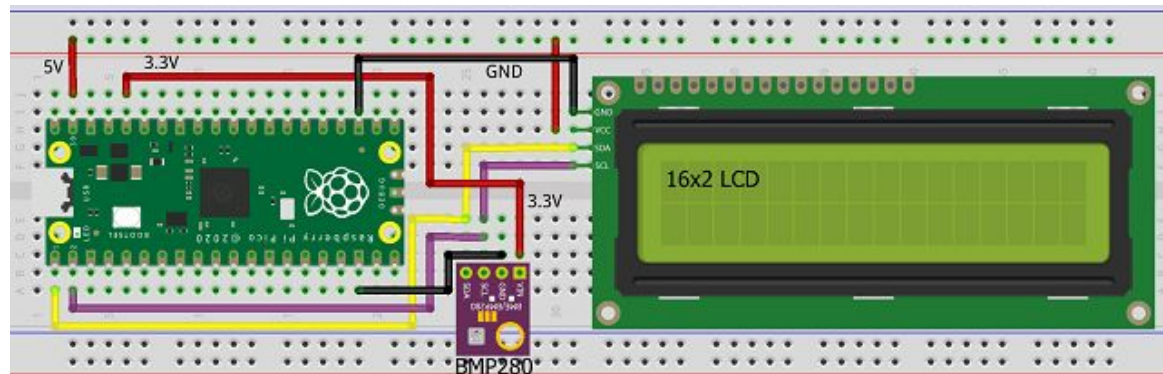
Monitoramento de temperatura e pressão



<https://how2electronics.com/interfacing-dht11-temperature-humidity-sensor-with-raspberry-pi-pico/>



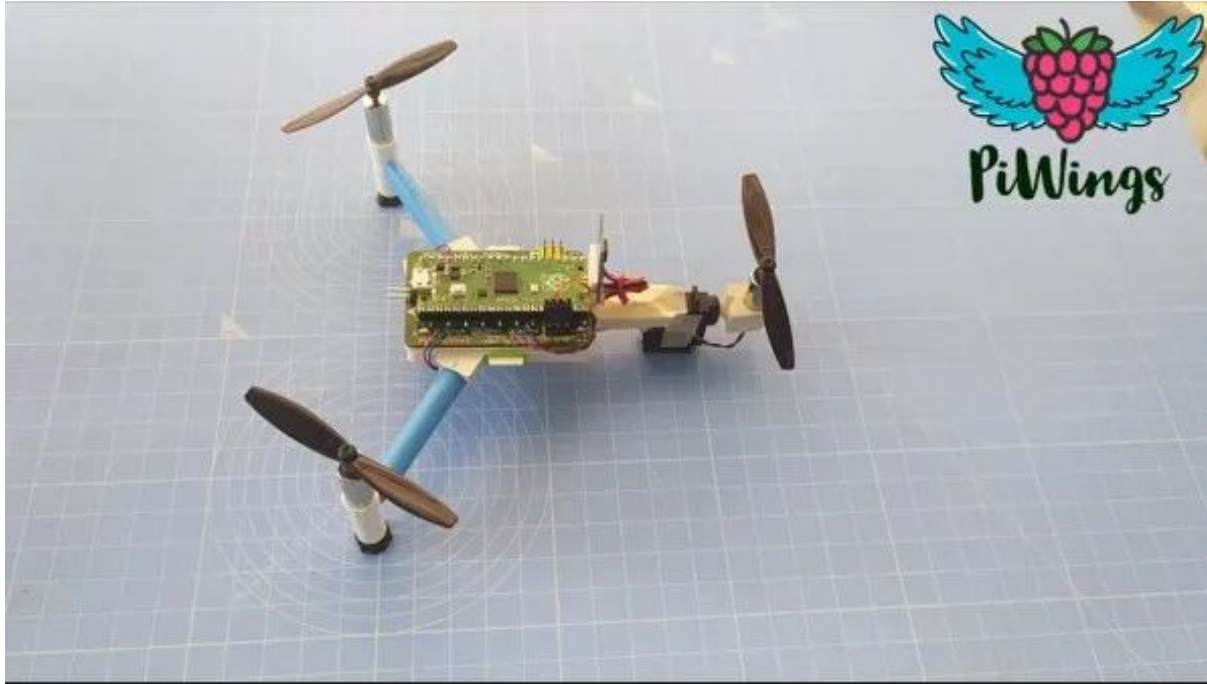
Controle meteorológico



Fonte:

<https://www.iotstarters.com/weather-station-with-bmp280-sensor-and-raspberry-pi-pico/>

Tricóptero baseado em Raspberry Pi Pico



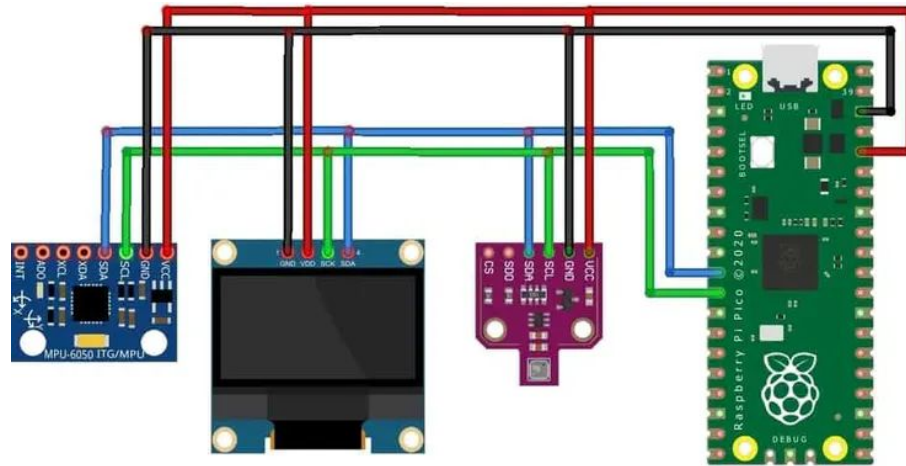
Vídeo do projeto: https://www.youtube.com/watch?v=0BZHFHRuc5s&ab_channel=ravibutani03

Controle de miniaturas



Vídeo do projeto: https://www.youtube.com/watch?v=tUmPXgipyKc&ab_channel=TheMotorChannel

Softwares para simulação



Página inicial - Wokwi

<https://wokwi.com/>



Página inicial - Fritzing

<https://fritzing.com/>

fritzing electronics
made easy

[Projects](#) [Parts](#) [Download](#) [Learning](#) [Services](#) [Contribute](#)

[FORUM](#) [FAB](#)



iOsnodente ▾

Login successful.



Fritzing is an **open-source hardware** initiative that makes electronics accessible as a creative material for anyone. We offer a software tool, a community website and services in the spirit of **Processing** and **Arduino**, fostering a creative ecosystem that allows users to **document** their prototypes, **share** them with others, **teach** electronics in a classroom, and layout and **manufacture** professional PCBs.

Download and Start

Download our **latest version 0.9.9** released on **September 24, 2021** and start right away.

Produce your own board

With **Fritzing Fab** you can easily and inexpensively turn your circuit into a real, custom-made PCB. Try it out now!

Participate

Fritzing can only act as a creative platform if many people are using it as a means of sharing and learning. **Let us know** how it fits your needs and how it doesn't, show it to your friends, and **share your**

[GO](#)

[FAQ](#) [ABOUT](#) [CONTACT](#)

Blog

Fritzing 0.9.9 released
September 24, 2021

Fritzing 0.9.8 released
August 09, 2021

PCBs at half the price
March 30, 2021

New Fritzing release 0.9.6
February 23, 2021

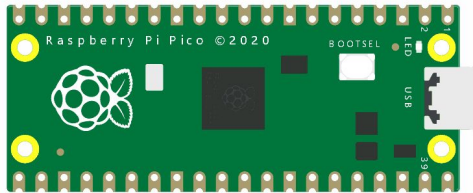
Fritzing maintenance release 0.9.4
February 13, 2021

[More posts...](#)

Propósito

Wokwi

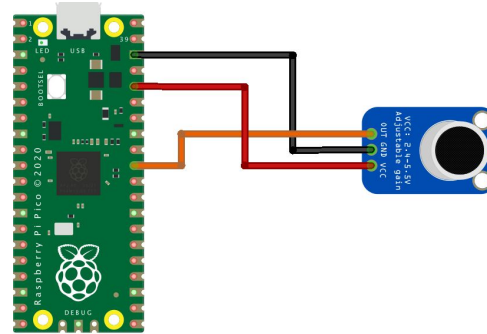
- Simulação online
- Simulações simples
- Não é escalável
- Para aprendizado
- Não permite upload do código



WOKWI

Fritzing

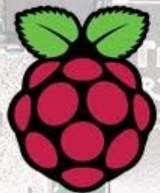
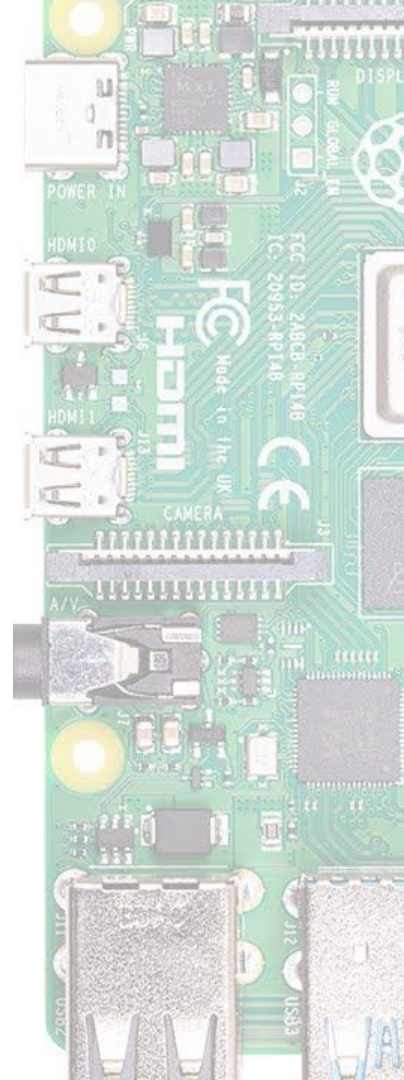
- Simulação offline
- Simulações complexas
- Escalável
- Para produção de projetos reais
- Permite upload do código



fritzing

Firmware

Programação do RP2040



Códigos para auxílio

Repositório GitHub - C/C++

<https://github.com/raspberrypi/pico-examples>

Repositório GitHub - Python

<https://github.com/raspberrypi/pico-micropython-examples>

Editores para Programação

Como programar em Python com Thony

[Programando o Raspberry Pi Pico em Python – Circuitaria](#)

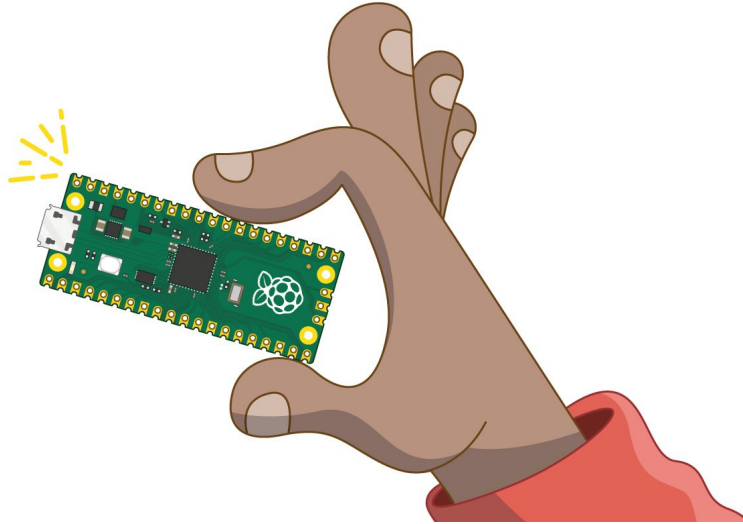
Como programar em C/Python com Vscode

[Programando a Raspberry Pi Pico no Visual Studio Code](#)

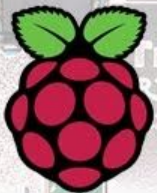
Como programar com a IDE do Arduino

<https://www.robocore.net/tutoriais/programacao-raspberry-pi-pico-arduino-ide>

Mãos na massa



Hello World Pico!!
Blink



IDE Arduino



Para usar: [Programação da Raspberry Pi Pico pela Arduino IDE](#)

Blink - Código comentado

```
Blink $
1
2 #define LED_BUILTIN 25 // Led construido na placa do Rasp
3 #define PIN_LED 13      // Led conectado em outra GPIO
4
5 void setup() {
6     // Função pinMode define a função da GPIO
7     // PIN_LED = número do pino que iremos configurar
8     // OUTPUT = função de SAÍDA do pino
9     pinMode(PIN_LED, OUTPUT);
10
11 }
12
13 void loop() {
14     // Função digitalWrite escreve o valor do pino
15     digitalWrite(PIN_LED, HIGH);
16     // delay = Pausa em mS -> 1000mS = 1S
17     delay(1000);
18
19     // HIGH = Nível alto = 3.3V
20     // LOW = Nível baixo = 0V
21     digitalWrite(PIN_LED, LOW);
22     // Mais uma pausa de 1S para manter o ciclo
23     delay(1000);
24 }
```



Verificar:



Conferir: Compilação terminada = OK

Compilação terminada.

Warning: Board attiny13:avr:attiny13e doesn't define a 'build.board' preference. Auto-set to: AVR_ATTINY13E

O sketch usa 78640 bytes (0%) de espaço de armazenamento para programas. O máximo são 16777216 bytes.

Variáveis globais usam 52656 bytes (19%) de memória dinâmica, deixando 217680 bytes para variáveis locais. O máximo são 270336 bytes.

Carregar:



Blink - Usando o Wokwi

WOKWI

SAVE

SHARE

Docs

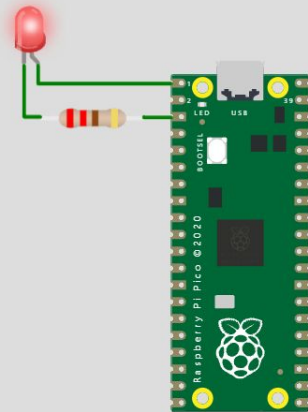
sketch.ino

diagram.json

```
1
2 #define LED_BUILTIN 25 // Led construido na placa do Rasp
3 #define PIN_LED 0      // Led conectado em outra GPIO
4
5 void setup() {
6     // Função pinMode define a função da GPIO
7     // PIN_LED = número do pino que iremos configurar
8     // OUTPUT = função de SAÍDA do pino
9     pinMode(PIN_LED, OUTPUT);
10
11 }
12
13 void loop() {
14     // Função digitalWrite escreve o valor do pino
15     digitalWrite(PIN_LED, HIGH);
16     // delay = Pausa em ms -> 1000ms = 1s
17     delay(1000);
18
19     // HIGH = Nível alto = 3.3V
20     // LOW = Nível baixo = 0V
21     digitalWrite(PIN_LED, LOW);
22     // Mais uma pausa de 1s para manter o ciclo
23     delay(1000);
24 }
25
26
27
28
29
30
31
32
```

Simulation

01:13.874 24%



Fade - IDE Arduino

Fade

```
1 #define PIN_LED 0 // Pino do LED
2 #define FATOR 5 // em quantos pontos aplicar o fade no LED
3
4 void setup() {
5     // Define o pino do LED como saída
6     pinMode(PIN_LED, OUTPUT);
7 }
8
9 void loop(){
10     // Criamos um laço que define a intensidade do brilho do LED
11     // Os parametros são: Condição inicial, condição de parada e fator de incremento
12     for ( int intensidade = 0; intensidade < 255; intensidade += FATOR ){
13         // A cada incremento do laço, aumentamos a intensidade do brilho
14         analogWrite( PIN_LED, intensidade );
15         delay(30);
16     }
17
18     // Fazemos o mesmo porém decrementando o valor do brilho
19     for ( int intensidade = 255; intensidade > 0; intensidade -= FATOR ){
20         analogWrite( PIN_LED, intensidade );
21         delay(30);
22     }
23 }
24
```

Fade - Usando o Wokwi

WOKWI

SAVE

SHARE

Docs

sketch.ino

diagram.json

```
1  #define PIN_LED 0 // Pino do LED
2
3  void setup() {
4      // Define o pino do LED como saída
5      pinMode(PIN_LED, OUTPUT);
6  }
7
8  void loop(){
9      // Criamos um laço que define a intensidade do brilho do LED
10     // Os parametros são: Condição inicial, condição de parada e fator de incremento
11     for ( int intensidade = 0; intensidade < 255; intensidade++){
12         // A cada incremento do laço, aumentamos a intensidade do brilho
13         analogWrite( PIN_LED, 200 );
14         delay(10);
15     }
16
17     // Fazemos o mesmo porém decrementando o valor do brilho
18     for ( int intensidade = 255; intensidade > 0; intensidade-- ){
19         analogWrite( PIN_LED, 100 );
20         delay(10);
21     }
22 }
23
24
25
26
27
28
29
30
31
32
```

Simulation

▶

+

⋮

