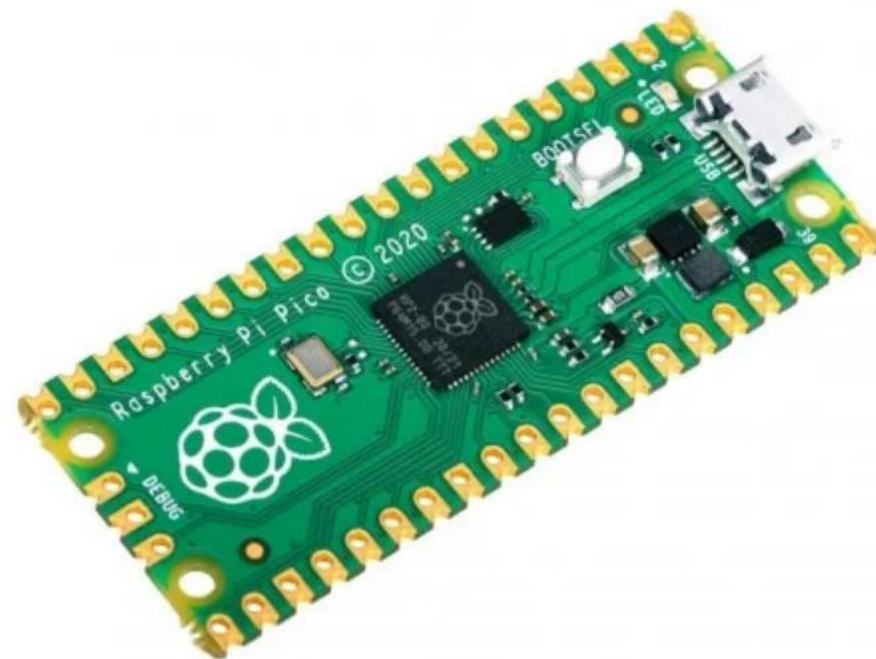


Introdução ao Raspberry Pi Pico

EEN251 - Microcontroladores e Sistemas Embarcados

Agenda

- Plataforma Raspberry Pi Pico;
- Linguagem MicroPython;
- Linguagem CircuitPython;
- Ambiente de Programação Thonny;
- Utilização do Raspberry Pi Pico;
- Mais informações.

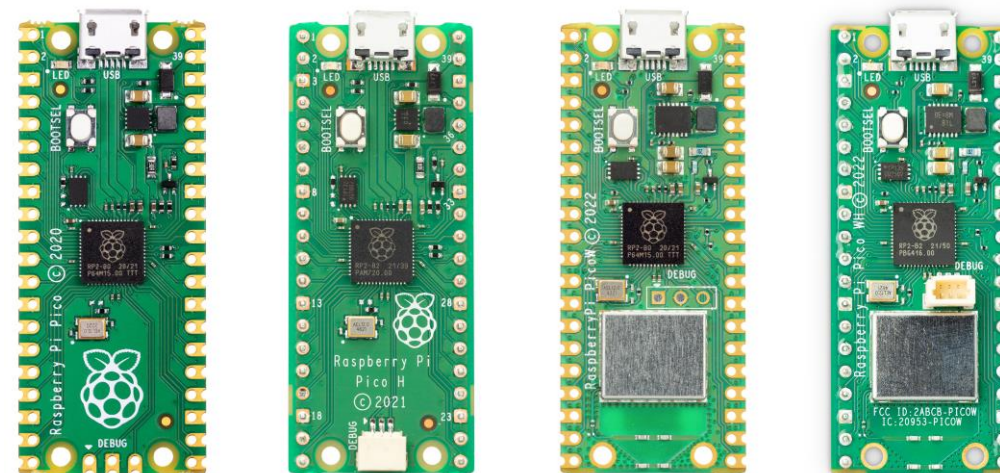


Plataforma Raspberry Pi Pico

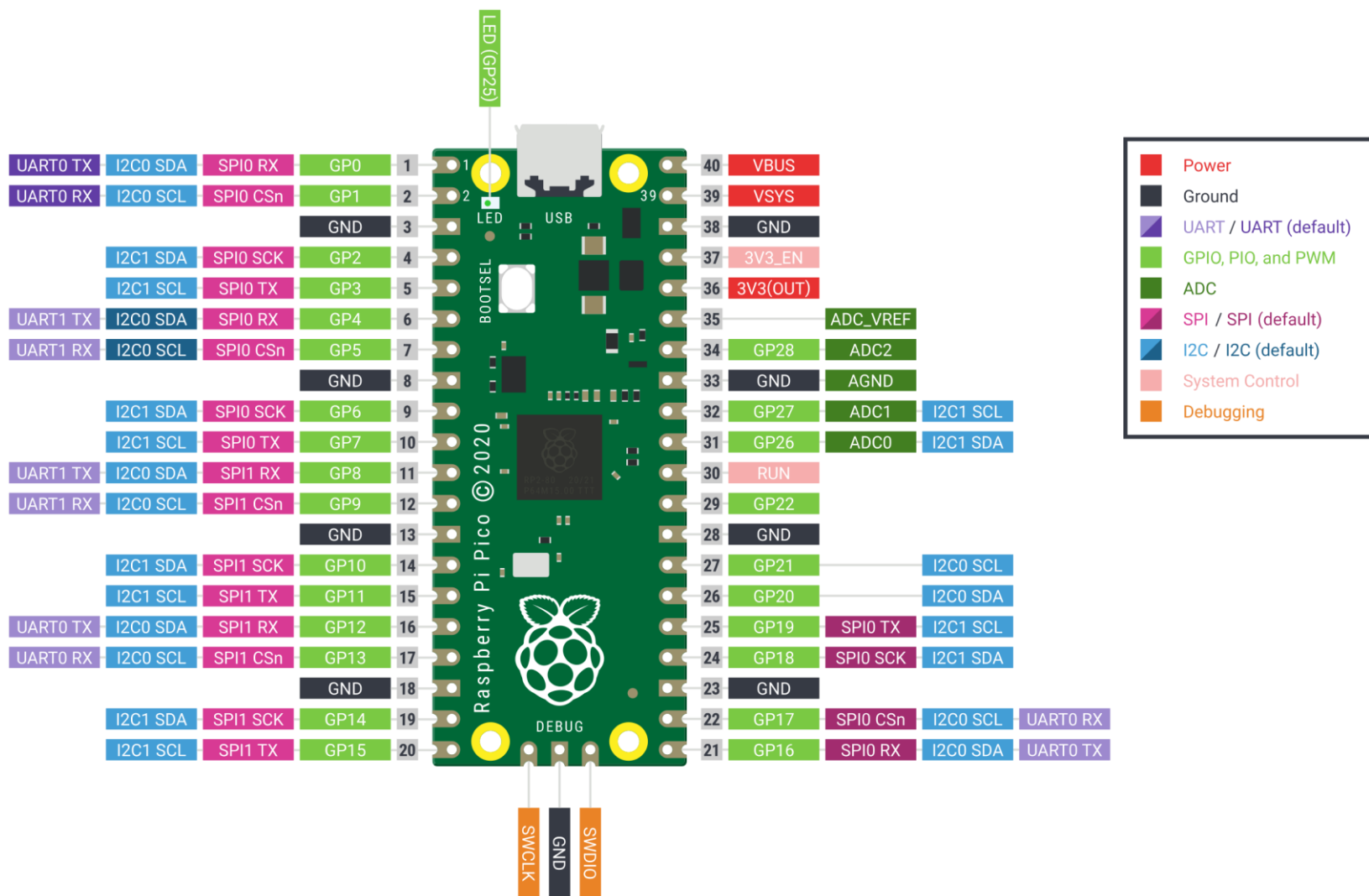
- O Raspberry Pi Pico é uma placa de microcontrolador de baixo custo e alta performance com interfaces digitais flexíveis. As principais características incluem:
 - Chip de microcontrolador **RP2040** projetado pela Raspberry Pi no Reino Unido;
 - Processador dual-core Arm Cortex M0+, com clock flexível de até 133 MHz;
 - 264 kB de SRAM e 2 MB de memória flash on-board;
 - USB 1.1 com suporte para dispositivo e host;
 - Modos de sleep e dormant de baixo consumo de energia;
 - Programação por arrastar e soltar usando armazenamento em massa sobre USB;
 - 26 pinos GPIO multifuncionais;
 - 2 × SPI, 2 × I2C, 2 × UART, 3 × ADC de 12 bits, 16 × canais PWM controláveis;
 - Relógio e temporizador precisos integrados;
 - Sensor de temperatura;
 - Bibliotecas de ponto flutuante aceleradas integradas;
 - 8 × máquinas de estado de I/O programáveis (PIO) para suporte a periféricos personalizados.

Plataforma Raspberry Pi Pico

- A Raspberry Pi suporta oficialmente para o **RP2040** as linguagens C e MicroPython. Também é possível usar outras linguagens, como o CircuitPython e Rust, mas essas não são suportadas oficialmente.
- A família Raspberry Pi Pico atualmente consiste em quatro placas:
 - Pico, apenas com processador **RP2040** e sem pinos;
 - Pico H, apenas com processador **RP2040** e com pinos soldados;
 - Pico W, com processador **RP2040** e módulo WiFi/Bluetooth e sem pinos;
 - Pico WH, com processador **RP2040** e módulo WiFi/Bluetooth e com pinos soldados.



Plataforma Raspberry Pi Pico

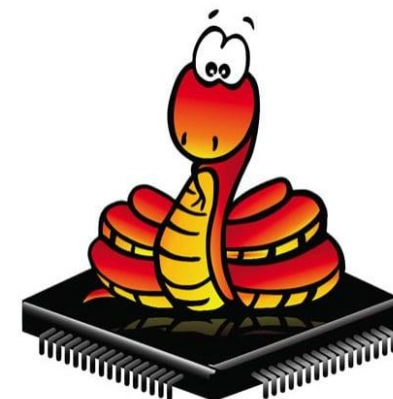


Plataforma Raspberry Pi Pico

- O Pico opera com 3,3V e pode ser alimentado de dois modos:
 - Pelo conector micro-USB com 5V. O pino **VBUS** pode ser usado para ter acesso aos 5V do micro-USB e fica zerado se o conector não for usado;
 - Pelo pino **VSYS**, que é a entrada principal de energia do sistema. A tensão de entrada pode variar entre 1,8V e 5,5V, permitindo que ele possa ser alimentado diretamente por baterias.
 - **Obs:** Não é recomendável utilizar o conector micro-USB ao mesmo tempo que uma tensão de alimentação no **VSYS**. Caso o **VSYS** seja utilizado, o ideal é remove-lo quando for utilizar o conector micro-USB para programação ou debug. Caso seja necessário usar duas fontes simultaneamente, é possível fazer uma proteção no **VSYS** com um diodo.
- O pino **3V3_EN** pode ser aterrado para desativar a fonte interna do Pico e desligá-lo.
- Por operar com 3,3V, não é recomendado conectar o Pico a entradas e saídas com 5V. Para esses casos, um conversor de nível lógico deve ser utilizado.

Linguagem MicroPython

- MicroPython é uma implementação enxuta e eficiente da linguagem de programação Python 3, que inclui um pequeno subconjunto da biblioteca padrão do Python e é otimizado para rodar em microcontroladores e ambientes com recursos limitados.
- É a linguagem padrão do Pico, e possui suporte para diversas outras plataformas (como ESP8266, ESP32, entre outras).
- Os recursos de hardware do Pico (GPIOs, PWM, ADC, DAC, UART, I2C, SPI, etc.) podem ser acessados através da biblioteca *machine*;
- Utiliza o formato de arquivo **.py**, igual ao Python.



Linguagem MicroPython

- O MicroPython suporta várias bibliotecas do Python, como:
 - *math*;
 - *random*;
 - *json*.
- Algumas bibliotecas do Python são portadas para o MicroPython com otimizações para trabalhar com plataformas embarcadas, como:
 - *utime*, similar a biblioteca *time* do Python, para trabalhar com temporização de eventos;
 - *ustruct*, similar a biblioteca *struct* do Python, para trabalhar com dados estruturados;
 - *_thread*, similar a biblioteca *threading* do Python, para trabalhar com tarefas e múltiplos núcleos.

Linguagem MicroPython

- Exemplos:
 - Piscar o LED do PICO em uma frequência de 0,5Hz;
 - Executar duas tarefas simultaneamente usando threads.

```

1 import machine
2 import utime
3
4 led = machine.Pin(25, machine.Pin.OUT)
5
6 while True:
7     led.value(1)
8     utime.sleep(1)
9     led.value(0)
10    utime.sleep(1)
11

```

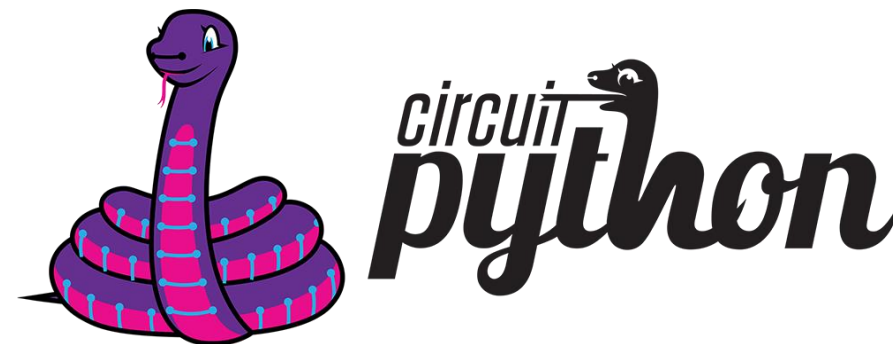
```

1 import _thread
2 import utime
3
4 def task1():
5     while True:
6         print("Executando tarefa 1")
7         utime.sleep(1)
8
9 def task2():
10    while True:
11        print("Executando tarefa 2")
12        utime.sleep(2)
13
14 _thread.start_new_thread(task1, ())
15 _thread.start_new_thread(task2, ())
16

```

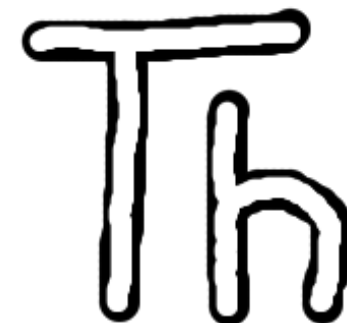
Linguagem CircuitPython

- Além do MicroPython, o Pico pode ser programado com o CircuitPython:
 - CircuitPython é a ramificação da Adafruit do MicroPython projetada para simplificar a experimentação e a educação em microcontroladores de baixo custo;
 - Muito similar ao MicroPython, mas com algumas diretrizes diferentes sobre como trabalhar com o hardware embarcado. Isso faz com que nem toda biblioteca criada para o CircuitPython funcione com o MicroPython;
 - Quando for usar uma biblioteca online, é necessário garantir que seja compatível com o MicroPython ou podem ocorrer problemas de execução;
 - Virtualmente todas as bibliotecas disponibilizadas pela Adafruit são escritas em CircuitPython.



Ambiente de Programação Thonny

- Thonny é um ambiente de desenvolvimento integrado (IDE) para Python que foi desenvolvido com o objetivo de tornar a programação em Python mais fácil e acessível para iniciantes;
- É o ambiente de programação padrão para o Raspberry Pico Pi, já possuindo suporte e ferramentas para utilizá-lo;
- A última versão pode ser baixada em: <https://thonny.org>.



Ambiente de Programação Thonny



Utilização do Raspberry Pi Pico

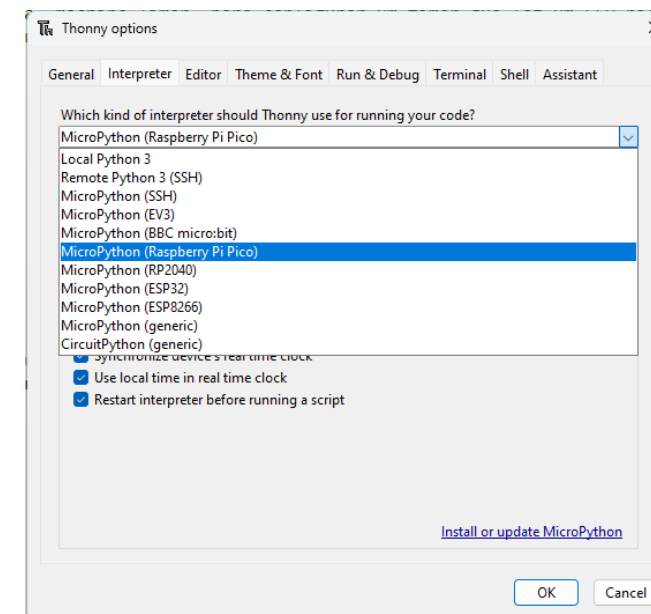
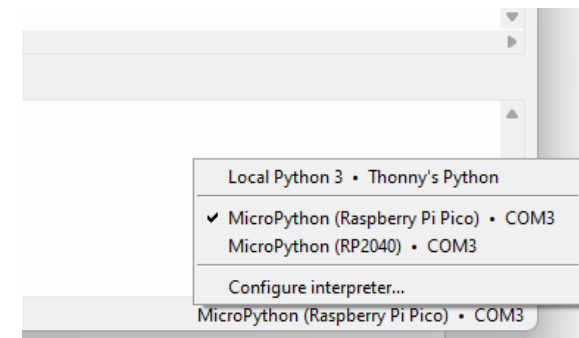
- Para utilizar o Pico com o Thonny, o MicroPython precisa estar instalado nele.
- O MicroPython pode ser facilmente instalado ou atualizado no Raspberry Pi Pico:
 - A versão mais recente dele pode ser baixada em: <https://micropython.org/download/rp2-pico/>;
 - Para instalar manualmente, o Pico deve ser conectado no computador com o botão **BOOTSEL** pressionado. Isso irá abrir o Pico como um disco no computador (com o nome **RPI-RP2**). O arquivo do MicroPython baixado deve ser arrastado para esse drive. O firmware será instalado e o Pico deve ser reinicializado para terminar o processo;
 - O MicroPython também pode ser instalado diretamente pelo Ambiente Thonny (instruções em <https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/3>).

Utilização do Raspberry Pi Pico

- Com o Thonny aberto e o Pico conectado ao computador, é possível selecionar a execução do interpretador no Pico:
 - No canto inferior direito do Thonny;
 - Ou então em “Tools >> Options... >> Interpreter”.
- Com o Pico conectado ao Thonny, você terá acesso ao REPL (Read Evaluate Print Loop), que permite à execução de instruções como se fosse um console Python.

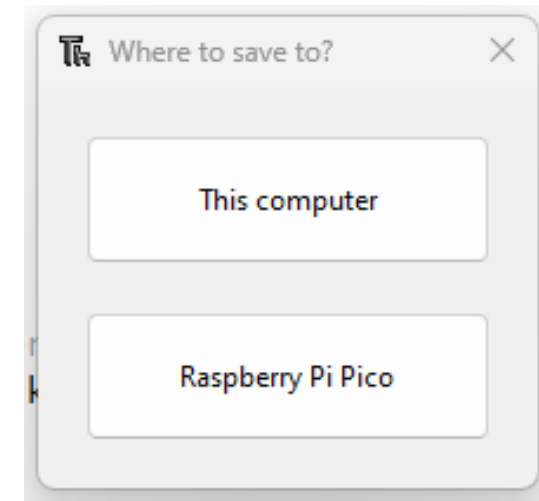
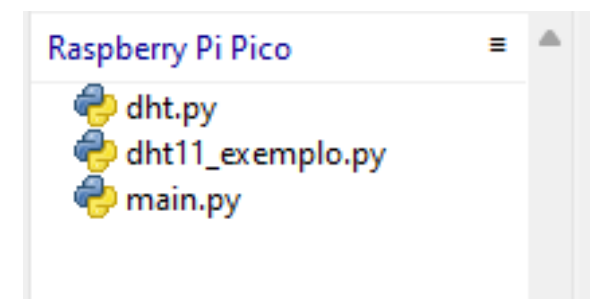
```

Shell x
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
    
```



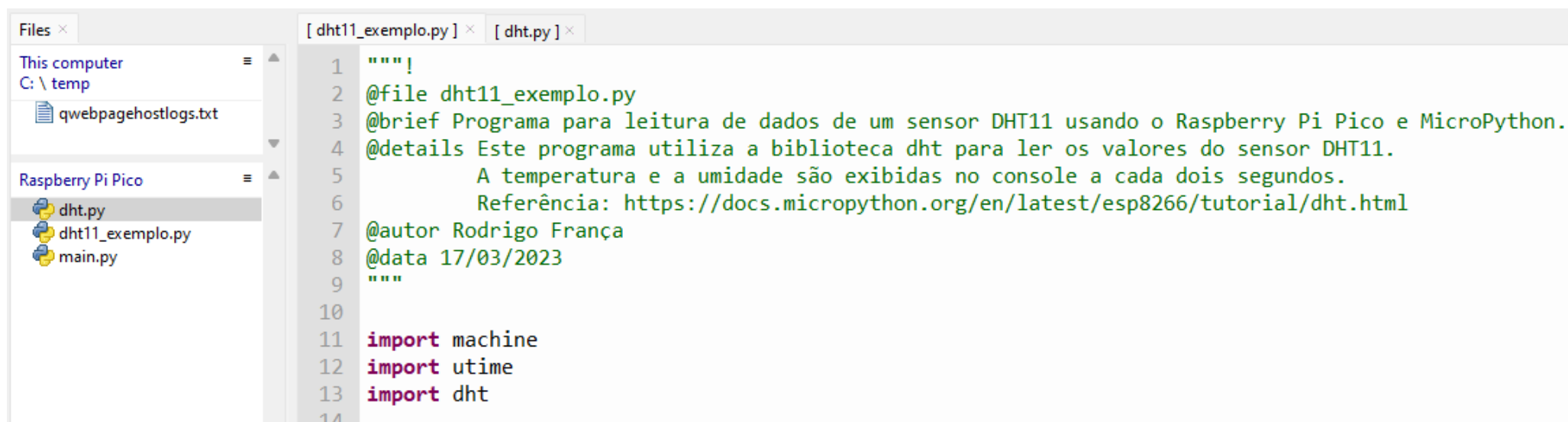
Utilização do Raspberry Pi Pico

- O Pico possui uma memória interna de 2 MB para guardar os arquivos em **.py**, assim como qualquer arquivo necessário para execução do código.
- Qualquer arquivo **.py** que precisa ser executado no Pico precisa estar necessariamente no seu armazenamento interno.
- O Thonny permite visualizar, editar e deletar os arquivos gravados internamente no Pico, assim como gravar novos arquivos. Quando a opção “File >> Save as...” é selecionada, você tem a opção de gravar o arquivo no computador ou no Pico.



Utilização do Raspberry Pi Pico

- Para utilizar bibliotecas externas no Pico, é necessário que todos os arquivos referentes a elas estejam no armazenamento interno dele;
- Quando o Pico é energizado, o primeiro arquivo que ele roda é o **main.py**. Logo, qualquer projeto que precise ser executado sem auxílio do Thonny precisa ter um **main.py**.



```

Files ×
This computer
C: \ temp
qwebpagehostlogs.txt

Raspberry Pi Pico
dht.py
dht11_exemplo.py
main.py

[ dht11_exemplo.py ] × [ dht.py ] ×
1  """!
2  @file dht11_exemplo.py
3  @brief Programa para leitura de dados de um sensor DHT11 usando o Raspberry Pi Pico e MicroPython.
4  @details Este programa utiliza a biblioteca dht para ler os valores do sensor DHT11.
5           A temperatura e a umidade são exibidas no console a cada dois segundos.
6           Referência: https://docs.micropython.org/en/latest/esp8266/tutorial/dht.html
7  @autor Rodrigo França
8  @data 17/03/2023
9  """
10
11  import machine
12  import utime
13  import dht
14

```

Utilização do Raspberry Pi Pico

- Diversos exemplos funcionais e bibliotecas para os sensores disponibilizados para o projeto da EEN251 podem ser encontrados no repositório https://github.com/rodmarfran/EEN251_Sistemas_Embarcados.

rodmarfran / EEN251_Sistemas_Embarcados Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags Go to file Add file <> Code

Your main branch isn't protected
Protect this branch from force pushing or deletion, or require status checks before merging. [Learn more](#) [Protect this branch](#)

rodmarfran Adicionado exemplo para o IMU 6050 e pequenas correções nos comentá... d7bc6b5 3 weeks ago 4 commits

Acelerometro e Giroscopio 6-DOF M...	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago
Display OLED	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago
Modulo Tiny RTC I2C	Organização dos exemplos em pastas e melhoria nos comentários de cód...	last month
Modulo Ultrassonico HC-SR04	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago
Sensor BMP280	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago
Sensor DHT11	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago
Sensor Fotoresistor LDR	Adicionado exemplo para o IMU 6050 e pequenas correções nos comen...	3 weeks ago

About
Repositório de códigos de exemplo para a matéria EEN251 - Sistemas Embarcados

Readme
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published

Utilização do Raspberry Pi Pico

- Exemplo de aplicação: pisca-pisca com LED do Pico usando interrupção de timer.

```
[ main.py ] ×  
1  """!  
2  @file main.py  
3  @brief Programa para fazer um LED piscar em um intervalo de tempo fixo usando o Raspberry Pi Pico e MicroPython.  
4  @details Este programa usa a biblioteca 'machine.Timer' para configurar um timer que faz um LED piscar em um intervalo de tempo fixo.  
5          Referência: https://projects.raspberrypi.org/en/projects/getting-started-with-the-pico/5  
6  @autor Rodrigo França  
7  """  
8  
9  import machine  
10  
11  # Configurando o LED e o Timer  
12  led = machine.Pin(25, machine.Pin.OUT)  
13  timer = machine.Timer()  
14  
15  # Função para fazer o LED piscar  
16  def blink(timer):  
17      led.toggle()  
18  
19  # Configurando o Timer para chamar a função blink em intervalos de tempo fixos  
20  timer.init(freq=2.5, mode=machine.Timer.PERIODIC, callback=blink)  
21
```

Mais informações

- Websites oficiais:
 - <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>;
 - <https://micropython.org>;
 - <https://circuitpython.org>;
 - <https://thonny.org>.
- Firmware atualizado do MicroPython (para o Raspberry Pi Pico):
 - <https://micropython.org/download/rp2-pico/>;
- Livro de referência e tutoriais para o Raspberry Pi Pico :
 - <https://hackspace.raspberrypi.com/books/micropython-pico/pdf/download>;
 - <https://www.robocore.net/tutoriais/programacao-raspberry-pi-pico-python>;
 - <https://projects.raspberrypi.org/en/projects/introduction-to-the-pico/>;
- Repositório GitHub com códigos de exemplo:
 - https://github.com/rodmarfran/EEN251_Sistemas_Embarcados.