Manual SCRAPY

Co-funded by the European Union

O apoio da Comissão Europeia à produção desta publicação não constitui um endosso do conteúdo, que reflete apenas as opiniões dos autores, e a Comissão não pode ser responsabilizada por qualquer uso que possa ser feito das informações nele contidas.

ADVERTÊNCIA

Ao utilizar este produto, tenha em atenção os seguintes pontos de aviso:

1. Este produto contém muitas peças pequenas. A deglutição ou o funcionamento inadequado podem causar infeções graves e morte. Procure assistência médica imediata quando o acidente acontecer.
2. Proibir estritamente o uso deste produto e suas partes perto de tomada elétrica CA ou outros circuitos em caso de risco potencial de choque elétrico.
3. Proibir estritamente o uso deste produto perto de qualquer líquido ou fogo.
4. Mantenha os materiais condutores longe deste produto.
5. Não permita que crianças com menos de 3 anos utilizem este produto sem a supervisão de um adulto. Por favor, coloque este produto numa posição em que as crianças com menos de 3 anos não possam chegar.
6. Não armazene ou use este produto em ambientes extremos, como calor ou frio extremo, alta umidade, sob luz solar direta, etc.
7. Lembre-se de desligar o circuito quando não for necessário.
8. Algumas partes deste produto podem tornar-se quentes ao toque quando usadas em determinados projetos de circuitos, o que é normal.
9. O uso inadequado pode causar superaquecimento.
10. A utilização de componentes não conformes com as especificações pode causar danos ao produto.

Introdução

O SCRAPY KIT é construído com base no uso do microcontrolador Raspberry Pi Pico. O "SCRAPY KIT" é criado no âmbito de um projeto cofinanciado pelo Erasmus+.

A nova tendência no ensino remoto, trazida pela pandemia de COVID-19, coloca uma lacuna no ensino e na prática de atividades com computação física atingindo alunos em risco de baixo desempenho nessas disciplinas STEM.

O objetivo do KIT é apoiar e promover a aprendizagem educacional prática em computação física e programação, em casos de ensino a distância e em sala de aula. O KIT fornece uma combinação de princípios de computação física e programação com hardware e software, levando a uma experiência de aprendizagem inovadora.

O presente Manual tem por objetivo:

- Informá-lo sobre os componentes do Kit e a utilização dos principais elementos eletrónicos.
- Guiá-lo passo a passo para montar efetivamente o KIT, considerando as precauções relacionadas.
- Fornecer tutoriais para o uso dos componentes e conexão com o microcontrolador Pico.
- Fornecer tutoriais com as funcionalidades e âmbito de cada componente eletrónico.

Divirta-se lendo e divirta-se através prática e experimentação com o KIT SCRAPY

Índice

Incluído no KIT SCRAPY

GPIO Breakout Board Raspberry Pi Pico (2pcs)

MB-102 módulo de fonte de alimentação (1pc)

White breadboard 830 pcs(1pc) + 400 pcs (1pc)

Botão de pressão (1pcs) & Tampa do botão (1pcs)

Campainha (1pc)

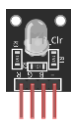
Resistências 220 Ohm (5 pcs) + 1k Ohm (5 pcs)

LDR fotoresistor (1pc)

100uF capacitor (1pc)

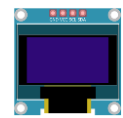
LED 3mm Azul (1pc), Verde (1pc), Vermelho(1pc) Amarelo (1pc)

RGB LED 5mm (1pc)

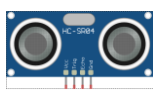


SG90 servo motor (1pc)

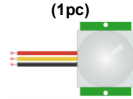
OLED I2C ICC (1pc)



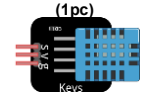
HC-SR04 sensor ultrassónico (1pc)



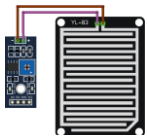
PIR Sensor Detetor de Movimento HC-SR501 (1pc)



DHT11 sensor digital de temperatura e humidade (1pc)

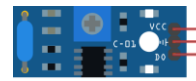


Sensor de chuva (1pc)

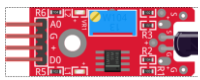


Potenciômetro Rotativo Linear B1k Ohm (1pc)

SW-420 sensor de vibração (1pc)

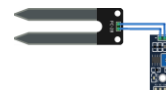


Sensor de calor (1pc)

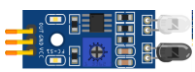


Sensor de deteção de som de alta sensibilidade (1pc)

Higrómetro de solo / sensor de deteção de humidade (1 pc)



Módulo de sensor infravermelho IR KY-032 (1pc)



Módulo de joystick (1pc)

Cabo USB para micro-USB 1m (1pc)

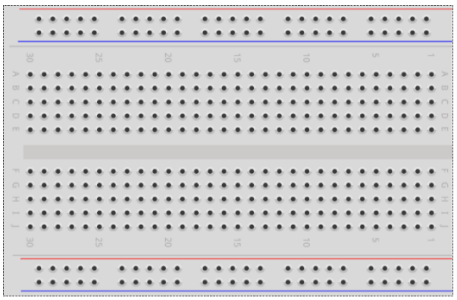
6 x AA 1.5V pilhas (1pc) + suporte (1pc)

Cabos jumper (6 pcs)

Parafusos (10 pcs) , porcas (6 pcs) e pilares de plástico (10 pcs)

Explicação dos componentes

1. O que é uma tábua de pão?



Uma placa de pão é uma placa de plástico com pequenos orifícios que permitem a fácil inserção de componentes eletrônicos (transístores, resistores, chips, etc.) para prototipar (construir e testar) um circuito eletrônico. O interior é composto por fileiras de pequenos grampos de metal para segurar os cabos a serem conectados.

A maioria das tábuas de pão tem fileiras de números, letras e sinais de mais e menos escritos nelas. O objetivo das etiquetas é ajudá-lo a localizar certos orifícios na placa de pão para que você possa seguir as instruções ao construir um circuito.

As tiras longas nos 2 lados da tábua de pão são geralmente marcadas com vermelho e azul ou vermelho e preto e com sinais de mais (+) e menos (-), respetivamente. Essas linhas são chamadas de barramentos ou trilhos e normalmente são usadas para fornecer energia elétrica ao circuito quando conectado a uma fonte de alimentação (bateria ou fonte externa).

O "lado" positivo está marcado em vermelho, tem o sinal de mais (+) e fornece a potência.

O "lado" negativo está marcado em azul ou preto, tem o sinal de menos (-) e fornece a ligação terra.

Vantagens de usar um Breadboard:

- Torna mais fácil verificar rapidamente circuitos simples e complexos e verificar facilmente circuitos na sua fase inicial.
- Fácil de ajustar.
- Flexível.
- Sem furos.
- Sem necessidade de soldadura.
- Fácil depuração de circuitos e programas.

2. O que é um resistor?

Um resistor é um pequeno pacote de resistência. Usá-lo em um circuito reduz a corrente em uma quantidade precisa. Para descobrir a resistência de um resistor existe um padrão de faixas coloridas.

Color	1st Band	2nd Band	3rd Band (5-Band Only)	Multiplier (3rd or 4th Band)	Tolerance (Last Band)
Black	0	0	0	1	
Brown	1	1	1	10	± 1%
Red	2	2	2	100	± 2%
Orange	3	3	3	1000	
Yellow	4	4	4	10000	
Green	5	5	5	100000	± 0.5%
Blue	6	6	6	1000000	± 0.25%
Violet	7	7	7	10000000	± 0.1%
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1	± 5%
Silver				0.01	± 10%
None					± 1%

(Crédito da imagem: Future Owns) disponível em https://www.tomshardware.com/how-to/resistor-color-codes

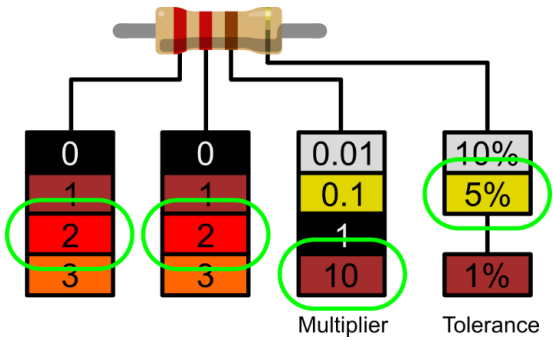
Códigos de cores de resistências comuns e seus usos:

Tipo de resistor	Código cores 4 bandas	Código cores 5 bandas	Uso mais comum
220Ohm	Vermelho-Vermelho-Castanho-Dourado	Vermelho-Vermelho-Preto-Preto-Dourado	Proteção de luz LED
1K Ohm (1Kiloohm)	Castanho-Preto-Vermelho-Dourado	Castanho-Preto-Preto-Castanho-Dourado	Proteção LED, divisor de tensão

As resistências não têm polaridade, por isso podem ser usadas em qualquer orientação num circuito. Mas para identificar os valores corretos do código de cores do resistor, precisamos entender as faixas coloridas no resistor.

Em um típico resistor de nível de hobby de quatro bandas, há três cores em um grupo. Estes são o primeiro, o segundo valor significativo e o multiplicador. A banda final é a tolerância do resistor, uma margem de erro, se quiser. Para a maioria dos amadores, uma tolerância de 5% (Ouro) é perfeita e comum.

Resistor de 220 Ohm (4 bandas)

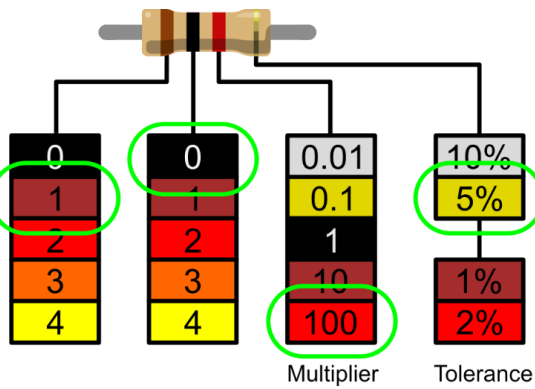


(Crédito da imagem: Future) disponível em https://www.tomshardware.com/how-to/resistor-color-codes

1. A primeira figura significativa é vermelha e usando o descodificador podemos ver que o vermelho tem um valor de 2.
2. O segundo número significativo também é vermelho, o que nos dá 22.
3. O multiplicador é marrom, e isso decodifica para 10. Se multiplicarmos 22 por 10, obtemos 220.
4. A faixa final, a tolerância, é o ouro. O ouro é de 5%, o que significa que podemos aceitar uma resistência com uma margem de erro de 5%.

Para fabricantes que exigem maior precisão, existem também cinco resistências de banda que têm um terceiro valor significativo. A figura extra fornece clareza, que pode ser essencial em circuitos sensíveis à resistência, por exemplo, instrumentos científicos e de engenharia.

1K Ohm Resistor (4-Band)



(Crédito da imagem: Tom's Hardware) disponível em https://www.tomshardware.com/how-to/resistor-color-codes

1. A 1ª linha é marrom, e usando o decodificador, podemos ver que o valor é 1.
2. A 2ª linha é preta, então isso nos dá 10.

3. O multiplicador é vermelho, e isso decodifica para 100. Se multiplicarmos 10 por 100 obtemos 1000.
4. A faixa final, tolerância, é ouro. O ouro é de 5%, o que significa que podemos aceitar uma resistência com uma margem de erro de 5%.

3. O que é um capacitor?

Um capacitor é um dispositivo que armazena energia elétrica em um campo elétrico. É um componente eletrônico passivo com dois terminais. Consiste em dois condutores elétricos que estão separados por uma distância. O espaço entre os condutores pode ser preenchido por vácuo ou com um material isolante conhecido como dielétrico. (Wikipédia)

O capacitor 100uF é um capacitor de desacoplamento eletrolítico. Esses capacitores são ótimos supressores de transientes/surtos e o uso de um entre a potência e o solo do circuito garante uma entrega de energia suave.

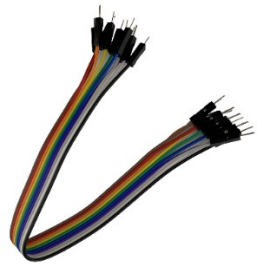
4. O que é um díodo?

Um diodo é um componente eletrônico de dois terminais que conduz a corrente principalmente em uma direção (condutância assimétrica); Tem baixa (idealmente zero) resistência em uma direção, e alta (idealmente infinita) resistência na outra.

A função mais comum de um diodo é permitir que uma corrente elétrica passe em uma direção (chamada de direção de avanço do diodo), enquanto a bloqueia na direção oposta (o inverso direção). Como tal, o díodo pode ser visto como uma versão

eletrónica de uma válvula de retenção. Este comportamento unidirecional é chamado de retificação e é usado para converter corrente alternada (ac) em corrente contínua (dc). Como retificadores, os diodos podem ser usados para tarefas como extrair modulação de sinais de rádio em recetores de rádio. (Wikipédia https://en.wikipedia.org/wiki/Diode)

5. O que é um cabo jumper?



Cabos jumper / fios são simplesmente fios que têm pinos de conector em cada extremidade, permitindo que eles sejam usados para conectar dois pontos um ao outro sem soldar. Os fios de jumper são normalmente usados com placas de pão e outras ferramentas de prototipagem, a fim de facilitar a mudança de um circuito, conforme necessário. A variação de cor dos fios pode ser usada como vantagem para diferenciar entre tipos de conexões, como terra ou energia.

Os fios de jumper normalmente vêm em três versões: macho para macho, macho para fêmea e fêmea para fêmea. A diferença entre cada um está no ponto final do fio. As extremidades masculinas têm um alfinete projetado e podem se conectar às coisas, enquanto as extremidades femininas não o fazem e são usadas para conectar as coisas. Os fios de jumper macho para macho são os mais comuns. Ao conectar duas portas em uma placa de pão, um fio macho para macho é mais necessário. (https://blog.sparkfuneducation.com/what-is-jumper-wire)

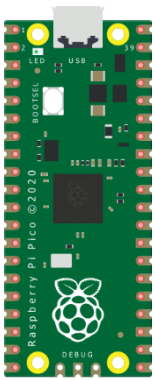
Preparação de Projetos

1. Leia-me antes de usar

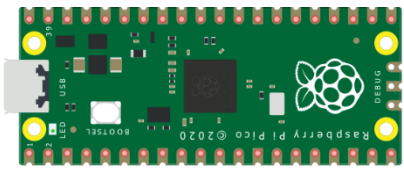
NOTA: Uma vez que as experiências envolvidas são todas experiências de circuito, uma ligação errada ou curto-circuito podem danificar a sua placa de desenvolvimento Pico. Por favor, verifique sempre o circuito novamente antes de ligar a fonte de alimentação.

2. Raspberry Pi Pico

Este é o Raspberry Pi Pico:

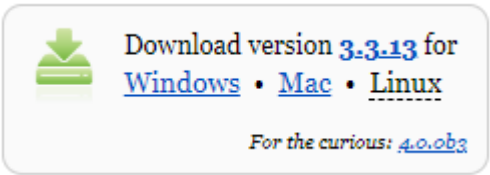


Ligue o cabo micro-USB à porta do lado esquerdo da placa.



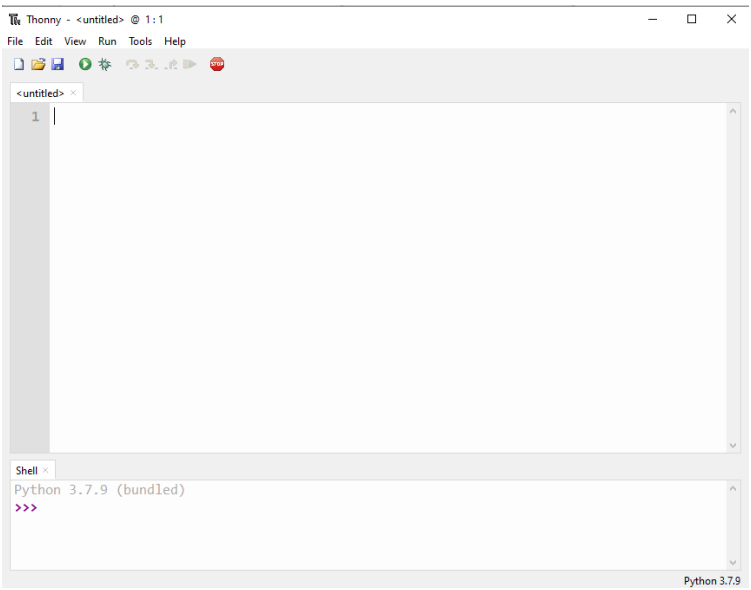
3. Instalar o Thonny IDE

Visite https://thonny.org e escolha o sistema operacional apropriado. Siga as instruções para concluir a instalação.



Neste manual, todos os tutoriais são programados no Windows 10, usando um Raspberry Pi Pico e o firmware apropriado.

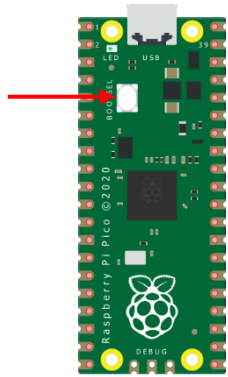
Após a conclusão da instalação, abra o Thonny a partir do seu computador.



4. Instalação de firmware

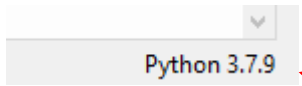
O Raspberry Pi Pico pode ser programado usando uma variante Python, chamada MicroPython. Para usar o MicroPython no Pico, primeiro você precisa instalar seu firmware.

Passo 1: Encontre o botão BOOTSEL no seu Raspberry Pi Pico.

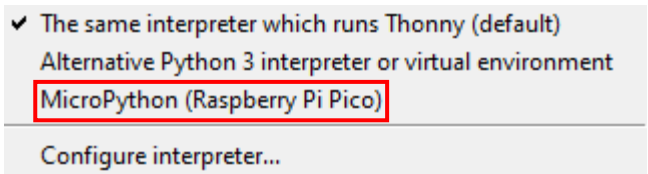


Passo 2: Pressione o botão BOOTSEL e segure-o enquanto conecta a outra extremidade do cabo micro-USB ao seu computador.

Passo 3: No canto inferior direito do Thonny você verá a versão do Python que você usa atualmente.

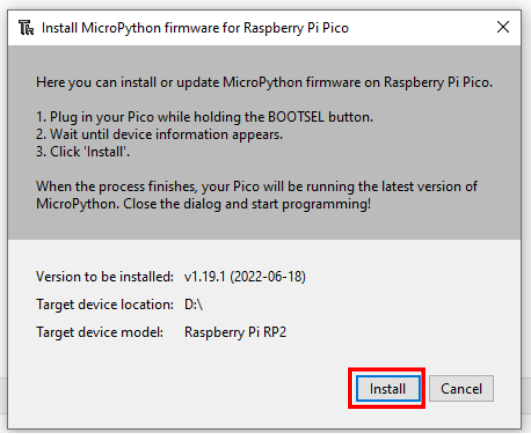


Clique na versão Python e escolha o MicroPython (Raspberry Pi Pico)

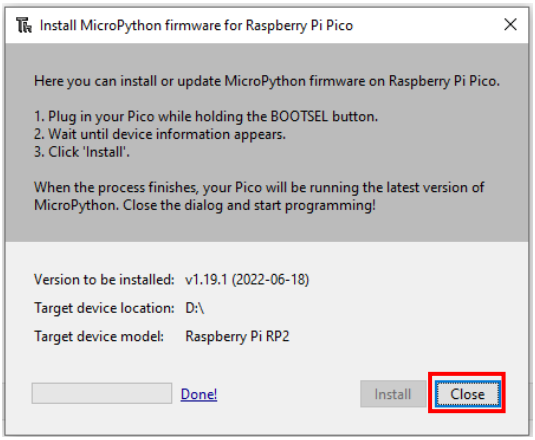


Se não vir esta opção, certifique-se de que o cabo está ligado corretamente no Pico e/ou no seu computador.

Passo 4: Irá aparecer uma caixa de diálogo, pedindo-lhe para instalar a versão de firmware mais recente no seu Pico. Clique no **botão Instalar** para copiar o firmware para o seu Pico.



Passo 5: Aguarde a conclusão da instalação e clique em **Fechar**.



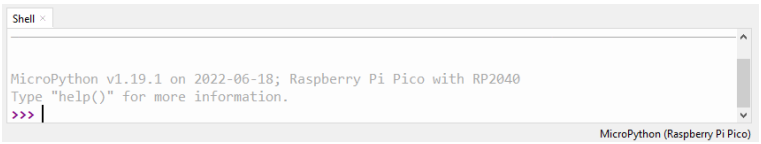
Você não precisa repetir o processo toda vez que conectar o Raspberry Pi Pico ao seu computador, então da próxima vez, basta conectá-lo e você está pronto para ir.

5. Introdução à Programação MicroPython

Agora vais usar o Thonny IDE para executar algum código Python simples para te familiarizares com o Shell e o MicroPython do Thonny.

Primeiro, certifique-se de que seu Raspberry Pi Pico está conectado ao seu computador e você selecionou o interpretador MicroPython conforme explicado na seção anterior.

O painel Shell na parte inferior do editor Thonny deve ter esta aparência:

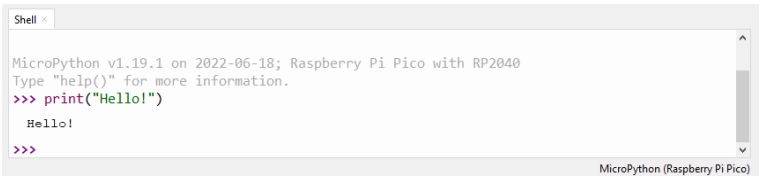


Thonny está pronto para se comunicar com seu Pico usando a estrutura REPL (loop de leitura-eval-impressão), que permite que você escreva código diretamente no Shell e obtenha saída.

Digite o seguinte comando:

```
print("Hello!")
```

Em seguida, pressione a tecla Enter e veja a seguinte saída:

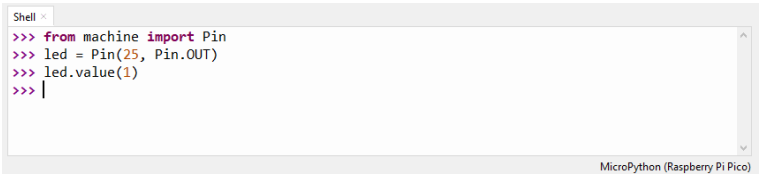


MicroPython allows you to add hardware-specific modules, such as `machine`, that you can use to program your Pico. In the no

exemplo seguinte, vais usar o módulo da `máquina` para ligar o LED integrado do Pico.

Escreva o seguinte código no Shell de Thonny:

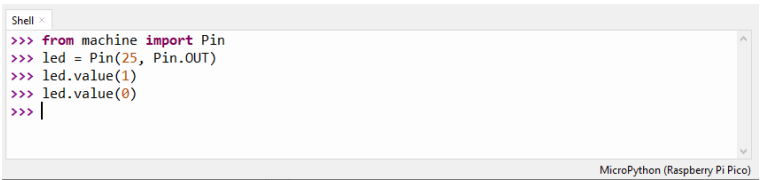
```
from machine import Pin
led = Pin(25, Pin.OUT)
led.value(1)
```



Pressione a tecla Enter e imediatamente o LED integrado do Pico se acenderá.

Para desligar o LED, escreva o seguinte código:

```
led.value(0)
```



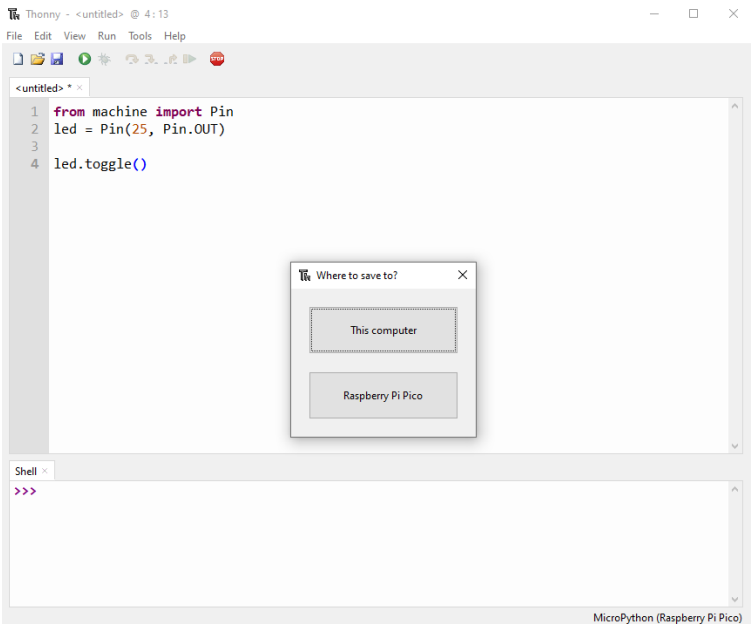
Para o resto desta seção, você escreverá seu primeiro programa "real" que fará o LED integrado piscar toda vez que você executar seu programa.

Thonny Shell é útil para experimentar comandos rápidos e certificar-se de que tudo está funcionando corretamente. No entanto, programas mais longos devem ser salvos em um arquivo .py. Usando Thonny, você pode salvar programas diretamente no Raspberry Pi Pico e, em seguida, executá-los.

Abra o Thonny Python e no painel principal do editor escreva o seguinte código:

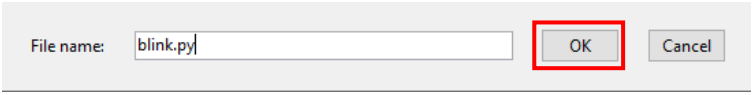
```
from machine import Pin
led = Pin(25, Pin.OUT)
led.toggle()
```

Agora, salve seu programa clicando no ícone Salvar no canto superior esquerdo ou pressionando Ctrl+S no teclado.



Thonny perguntará onde você quer que seu programa seja salvo. Escolha o **Raspberry Pi Pico**. Salve o arquivo como *blink.py* e clique em **OK.** Você sempre precisa adicionar a

extensão .py para que Thonny reconheça o arquivo como um arquivo Python.



Agora, sempre que clicar no ícone Reproduzir, deverá ver o LED integrado a ligar e a desligar.

Levando seu código um passo adiante, você pode fazer o LED integrado piscar em um determinado ritmo.

Escreva o seguinte código e salve seu programa usando o mesmo nome acima.

```
from machine import Pin
from time import sleep
led = Pin(25, Pin.OUT)
while True:
      led.toggle()
      sleep(1)
```

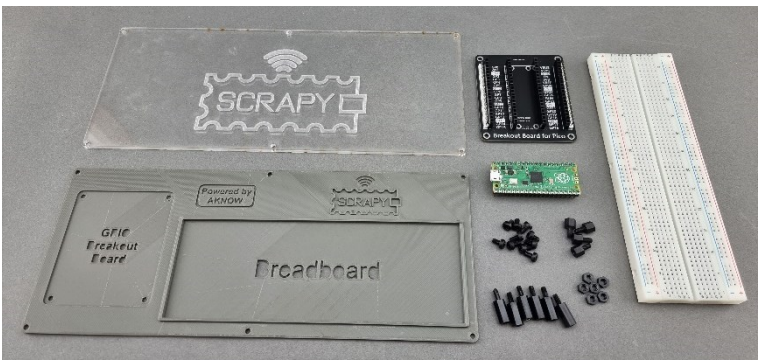
Agora, quando você executar o programa, o LED integrado piscará a cada 1 segundo até pararmos o programa. Para parar o programa, você pode clicar no ícone STOP ou pressionar Ctrl+C no teclado.

Em tutoriais futuros, aprenderemos a adicionar e controlar outros eletrônicos e sensores e criar programas que podem se comunicar uns com os outros.

Montagem do Kit

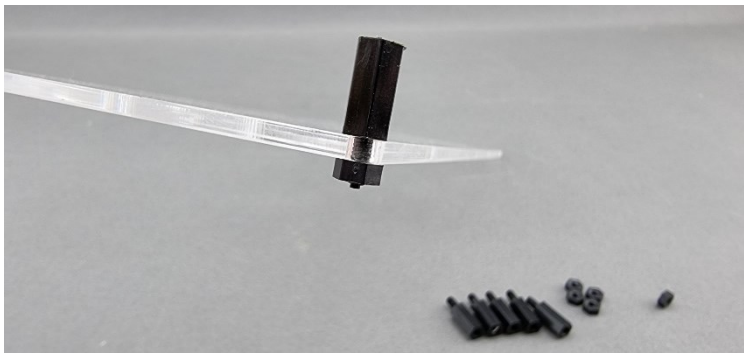
O Kit SCRAPY é composto pelos seguintes itens:

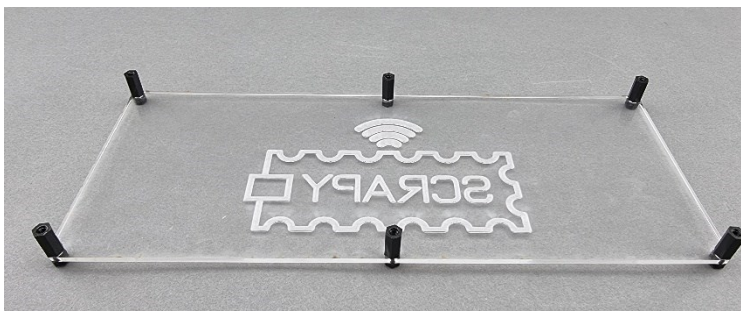
- 1x peça impressa em 3D
- 1 x peça de plexiglass
- 1 x Raspberry Pi Pico
- 1 x placa de breakout GPIO
- 1 x Breadboard (830pcs)
- 10 x parafusos de plástico
- 6 pilares de plástico de 12 mm
- 4 pilares de plástico de 6 mm
- 6 x porcas plásticas



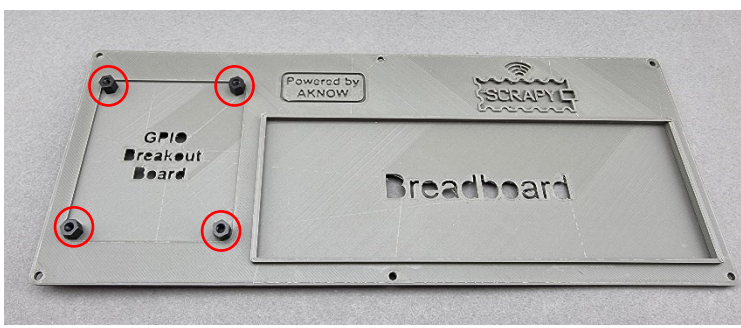
O procedimento de montagem é simples e pode ser concluído em 6 passos:

Passo 1: Monte os pilares de plástico 6 x 12mm com as 6 porcas de plástico na peça de plexiglass.

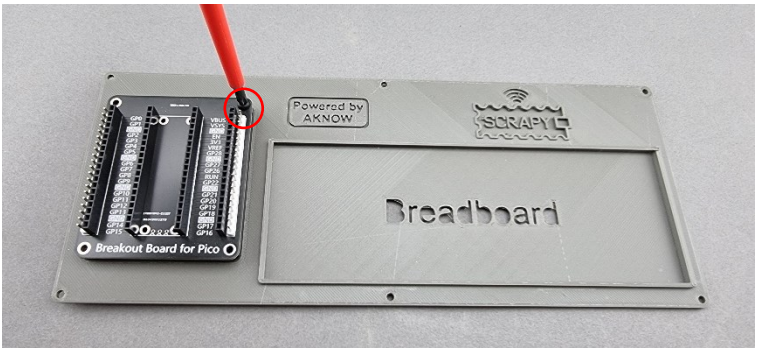


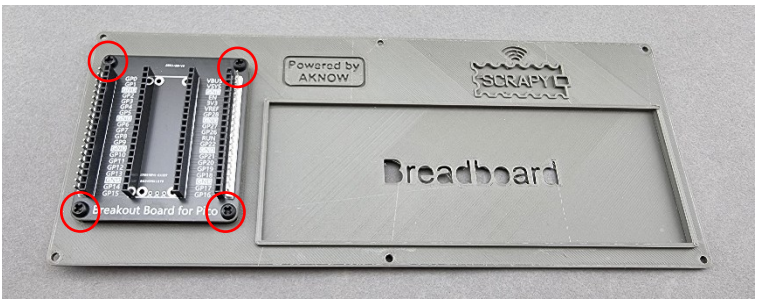


Passo 2: Na peça impressa em 3D, coloque 4 pilares de plástico x 6mm na seção "GPIO Breakout Board".

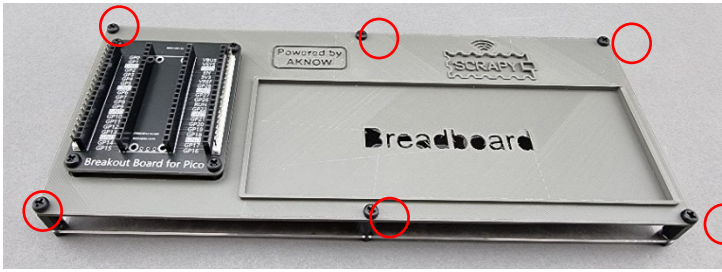


Passo 3: Monte a placa de breakout GPIO nos 4 pilares usando 4 parafusos plásticos.

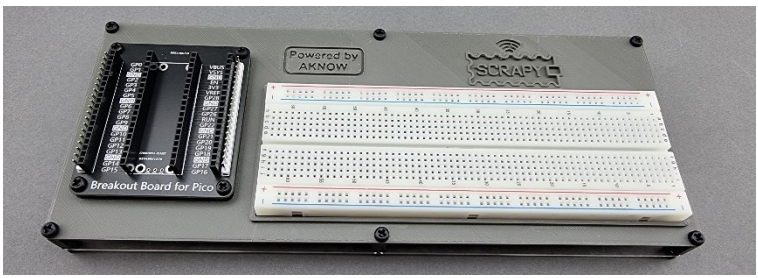




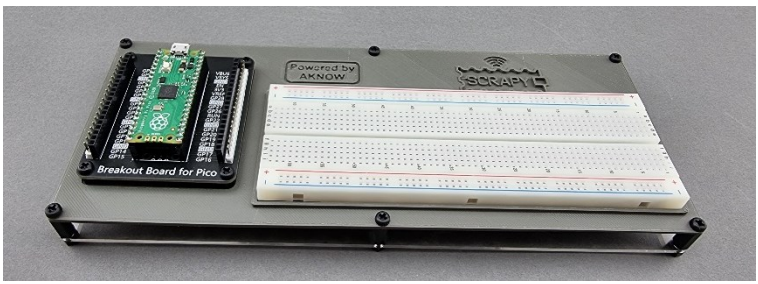
Passo 4: Monte o objeto impresso em 3D com o objeto de plexiglass usando 6 parafusos de plástico nos 6 pilares de plástico que você montou no Passo 1.



Passo 5: Tire a fita protetora por baixo do breadboard de 830 pcs e coloque-o na seção "Breadboard" do Kit. Por favor, certifique-se de que o lado positivo (+) do breadboard está no topo, como indica a imagem a seguir.



Passo 6: Coloque o Raspberry Pi Pico em cima da placa de breakout GPIO e pressione contra ele até que todos os pinos GPIO tenham sido inseridos corretamente. Certifique-se de que a ranhura micro-USB está na parte superior, como indica a imagem.



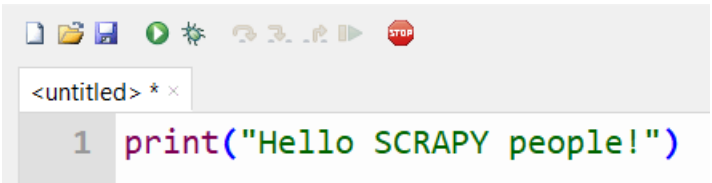
Tutoriais básicos

0. “Hello SCRAPY people!”

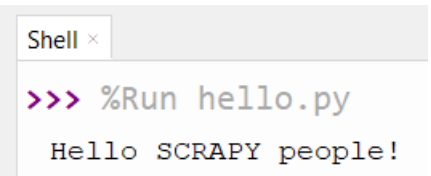
Neste tutorial básico, vamos aprender como imprimir uma mensagem simples em Thonny usando programação Python.

A função print()

1. Vá para o editor Thonny, escreva o seguinte código e pressione o ícone de reprodução. Thonny irá pedir-lhe para salvar o seu programa primeiro. Salve-o com o nome hello.py.

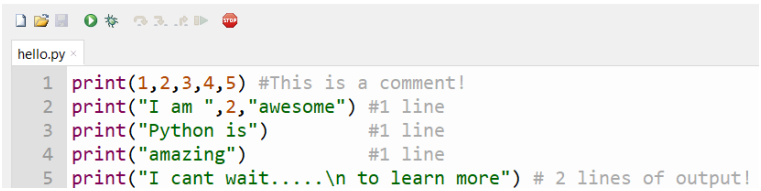


2. Verifique a janela do shell.

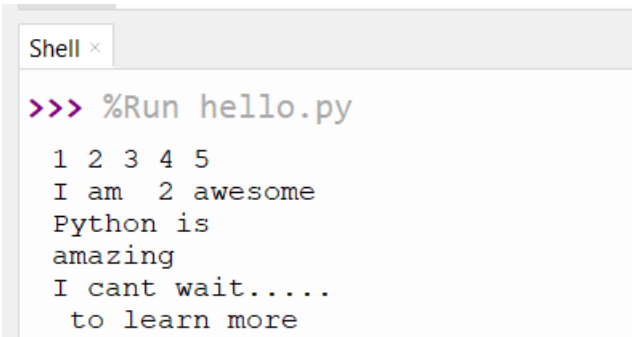


3. Muito bom trabalho! Você acabou de criar seu primeiro programa Python.

A função de impressão é uma função Python integrada que nos permite imprimir texto no shell. Também pode ter parâmetros. Crie um programa e copie o código a seguir, pressione play e veja como o texto aparece no shell.



Como você pode ver na janela do shell, cada função de impressão imprime texto em uma linha separada. No entanto, se você usar o "\n" (caractere de nova linha), poderá alterar a linha na mesma instrução de impressão.



Exercício

Use a função de impressão para imprimir em 3 linhas separadas "Adeus pessoas SCRAPY!" usando apenas uma instrução de impressão.

1. Controle um LED

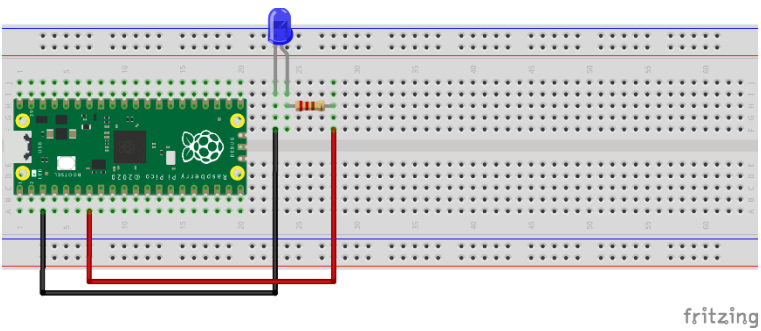
Descrição

Neste tutorial, você aprenderá como conectar e controlar uma luz LED. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `led.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x cabo Micro-USB
- 1 x Kit breadboard Pico
- 2 x Fios jumper macho/macho
- 1 x 220 resistor Ohm

Diagrama de ligações



– conecte a extremidade mais longa (+) do LED a uma resistência de 220Ohm

– conecte o resistor ao GPIO5 (cabo vermelho)

– conecte a extremidade mais curta (-) do LED a um pino GND

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /led.py @ 9 : 1

File Edit View Run Tools Help

[led.py]

```
1  from machine import Pin #import  Pin from library machine
2  from time import sleep  #import sleep from library time
3
4  led = Pin(5, Pin.OUT)
5
6  while True:
7      led.toggle()
8      sleep(1)
9
```

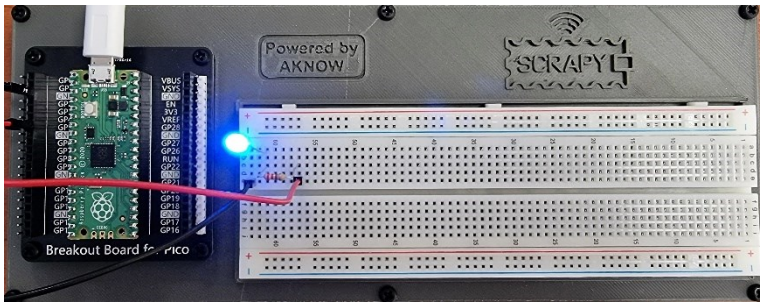
Shell

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

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o circuito parece usando o hardware fornecido:



2. Botão de pressão

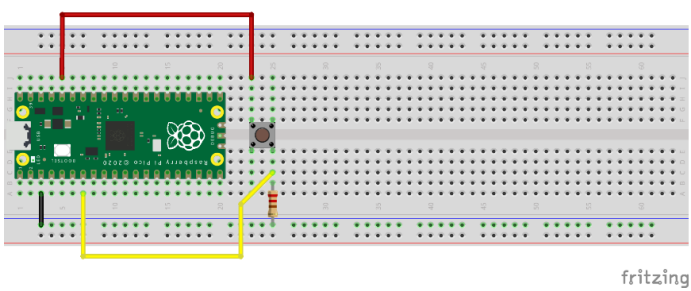
Descrição

Neste tutorial, você aprenderá como conectar e controlar um botão de pressão. Abra Thonny Python e, em seguida, vá para File → Save as..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `button.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Botão (qualquer cor)
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho/macho
- 1 x 220 Resistor ohm
- 1 x Tampa do botão

Diagrama de ligação



– Conecte o lado superior esquerdo do botão ao pino 3V3 (cabo vermelho)

– conecte o lado inferior direito do botão ao GPIO5 (cabo amarelo)

– conecte um pino GND ao trilho (-) (cabo preto)

– conecte o resistor de 220 Ohm ao trilho (-) e ao lado inferior direito do botão

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /button.py @ 10 : 1

File Edit View Run Tools Help

[button.py]

```
1  from machine import Pin #import Pin and ADC from library machine
2  from time import sleep  #import sleep from library time
3
4  button = Pin(5, Pin.IN, Pin.PULL_DOWN)
5
6  while True:
7      if button.value():
8          print("Button is pressed!")
9          sleep(1)
10
```

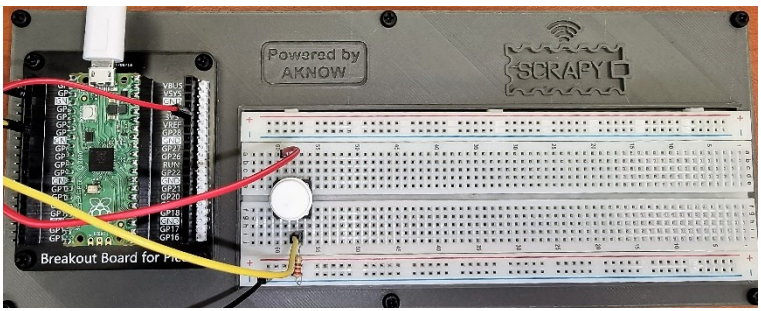
Shell

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

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o circuito parece usando o hardware fornecido:



3. Campaínha

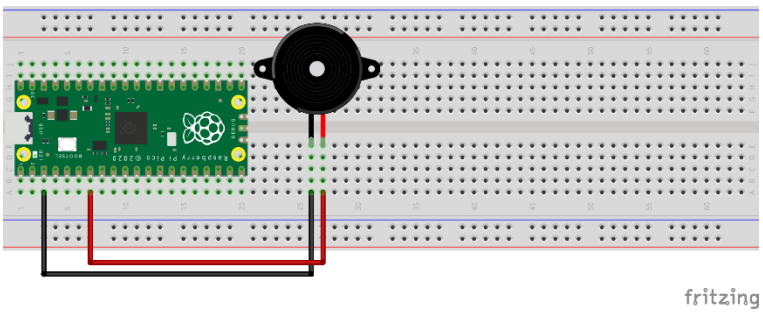
Descrição

Neste tutorial, você aprenderá como conectar e controlar uma campainha. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `buzzer.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 2 x Cabo jumper macho/macho
- 1 x Campaínha

Diagrama de ligação



– conecte a extremidade mais longa (+) da campainha ao pino GPIO5

– conectar a extremidade mais curta (-) da campainha a um pino GND

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /buzzer.py @ 16 : 1

File Edit View Run Tools Help

[buzzer.py]

```
1  from machine import Pin #import Pin from library machine
2  from time import sleep  #import sleep from library time
3
4  buzzer = Pin(5, Pin.OUT)
5
6  buzzer.value(0)
7  sleep(0.5)
8  buzzer.value(1)
9  sleep(0.5)
10 buzzer.value(0)
11 sleep(0.5)
12 buzzer.value(1)
13 sleep(0.5)
14 buzzer.value(0)
15 sleep(0.5)
16
```

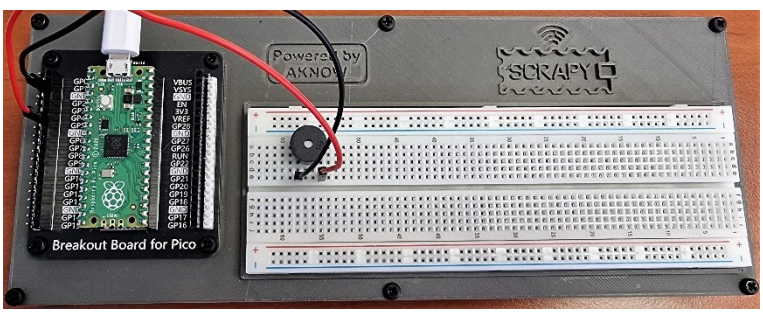
Shell

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

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o circuito parece usando o hardware fornecido:



4. Potenciómetro

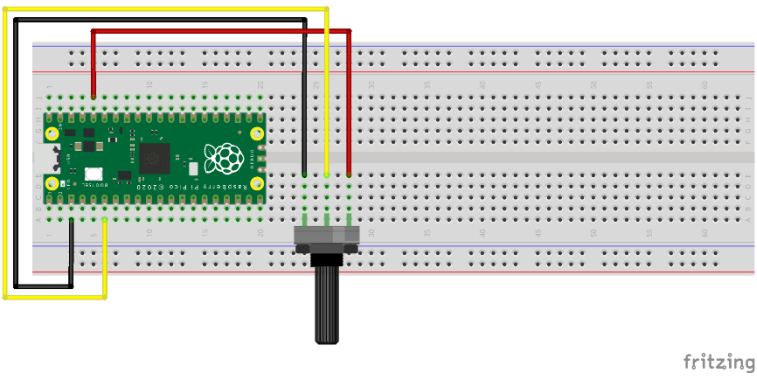
Descrição

Neste tutorial, você aprenderá como conectar e controlar o potenciômetro. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `pot.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 2 x Cabo jumper macho/macho
- 1 x Potenciómetro

Diagrama de ligação



– cabo preto deve ser conectado ao GND (pino 3) pino

– cabo amarelo deve ser conectado ao pino GPIO5

– cabo vermelho deve ser conectado ao pino de alimentação 3V3

– Rode o potenciómetro para a esquerda para que fique desligado

Código

Código MicroPython para o tutorial:

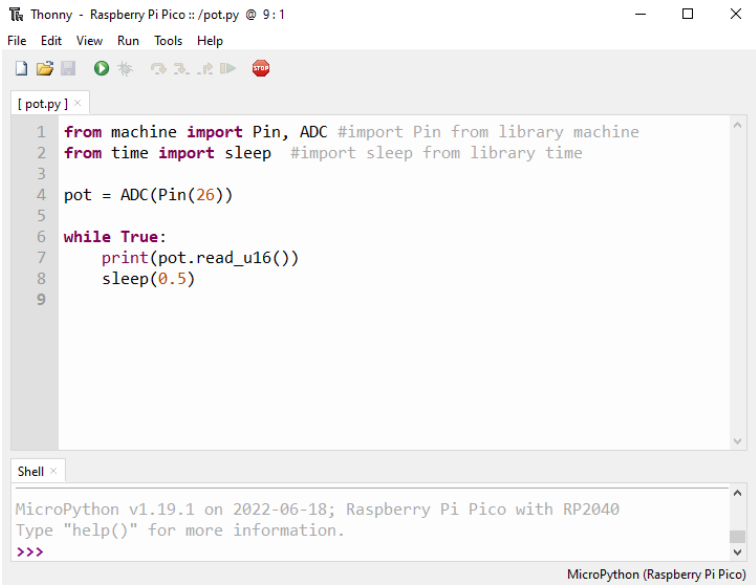
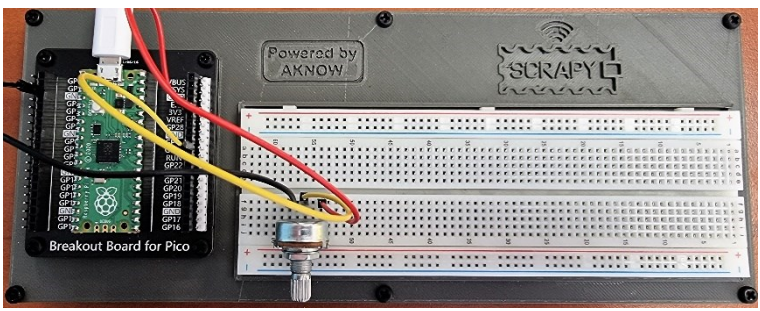


Imagem de exemplo

Imagem de como o circuito parece usando o hardware fornecido:



8. LED RGB

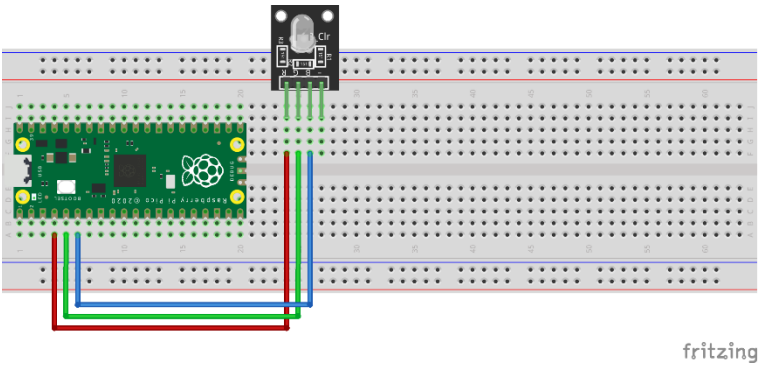
Descrição

Neste tutorial, você aprenderá como conectar e controlar uma luz de módulo LED RGB. Abra o Thonny Python e, em seguida, vá para Salvar → arquivo como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `rgb.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 4 x Cabo jumper macho-macho
- 1 x Módulo LED RGB

Diagrama de ligação



– cabo vermelho deve ser conectado ao GPIO2

– cabo verde deve ser conectado ao GPIO3

– cabo azul deve ser conectado ao GPIO4

– nenhuma conexão GND é necessária

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /rgb.py @ 14 : 1

File Edit View Run Tools Help

[rgb.py]

```
from machine import Pin, PWM #import Pin from library machine
from time import sleep  #import sleep from library time
import random
Led_R = PWM(Pin(2))
Led_G = PWM(Pin(3))
Led_B = PWM(Pin(4))
# Define the frequency
Led_R.freq(2000)
Led_G.freq(2000)
Led_B.freq(2000)
while True:
    # range of random numbers
    R=random.randint(0,65535)
    G=random.randint(0,65535)
    B=random.randint(0,65535)
    print(R,G,B)
    Led_R.duty_u16(R)
    Led_G.duty_u16(G)
    Led_B.duty_u16(B)
    sleep(1)
```

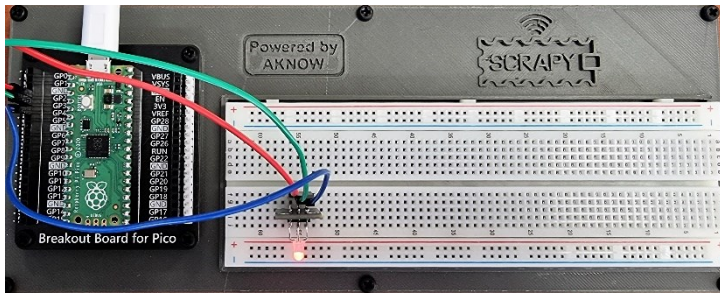
Shell

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

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o circuito parece usando o hardware fornecido:



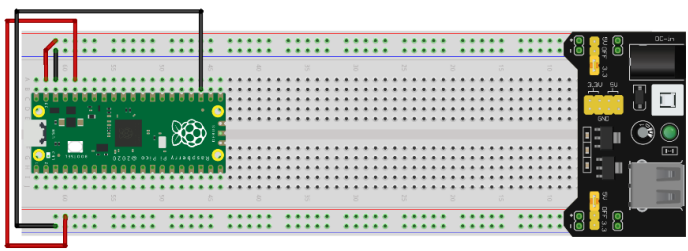
Tutoriais Avançados

Antes de prosseguir para esta seção e para a próxima, precisamos fazer algumas modificações em nosso Kit SCRAPY. Isso envolve a adição de alguns componentes extras e sua conectividade.

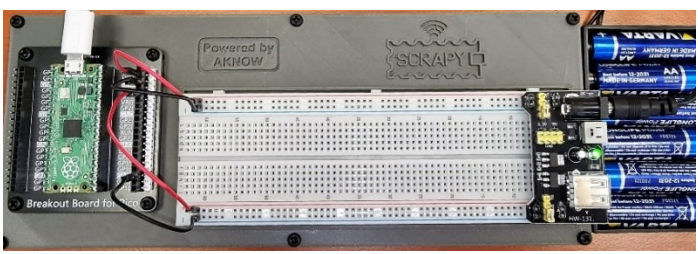
Componentes

- 1 x 6-AA Pack de pilhas
- 1 x MB-102 módulo de alimentação
- 4 x Cabos jumper macho-macho

Siga o esquema para conectar os novos componentes:



Circuito



– esta configuração será usada para os tutoriais restantes

– ligações laterais superiores: VSYS 5V ((+) vermelho) e GND ((-) preto)

– bottom side connections: 3V3 ((+) red) and GND ((-) black)

5. Fotoresistor LDR

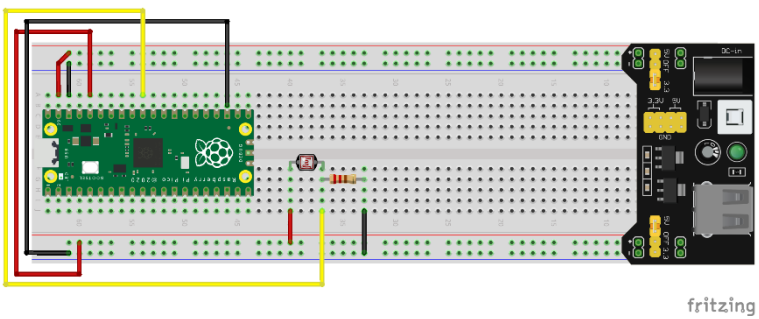
Descrição

Neste tutorial, você aprenderá como conectar e controlar um fotoresistor LDR. Abra Thonny Python e, em seguida, vá para Salvar → arquivo como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `ldr.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x 220Ohm resistor
- 1 x Kit Breadboard Pico
- 3 x Cabo jumper macho-macho
- 1 x Fotoresistor LDR

Diagrama de ligação



– o lado esquerdo do LDR está ligado ao trilho de 3V ((+) vermelho)

– o lado direito do LDR está ligado a uma resistência de 220 Ohm e ao pino GPIO26 ADC (cabo amarelo)

– o lado direito do resistor está ligado ao trilho GND ((-) preto)

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /ldr.py @ 10 : 1

File Edit View Run Tools Help

[ldr.py]

```
1  from machine import Pin, ADC #import Pin and ADC from library machine
2  from time import sleep  #import sleep from library time
3
4  ldr = ADC(Pin(26))
5
6
7  while True:
8      print(ldr.read_u16())
9      sleep(1)
10
```

Shell

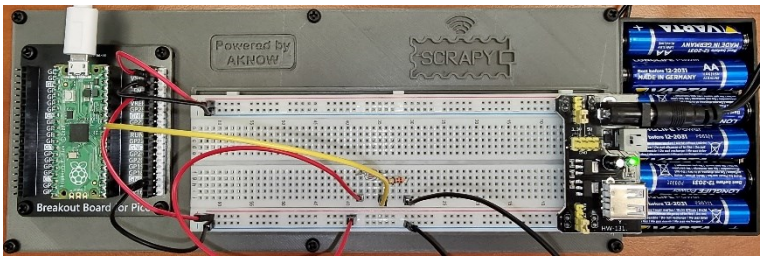
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  8642
  7329
  4032
  3504
  3840
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



6. Servo motor

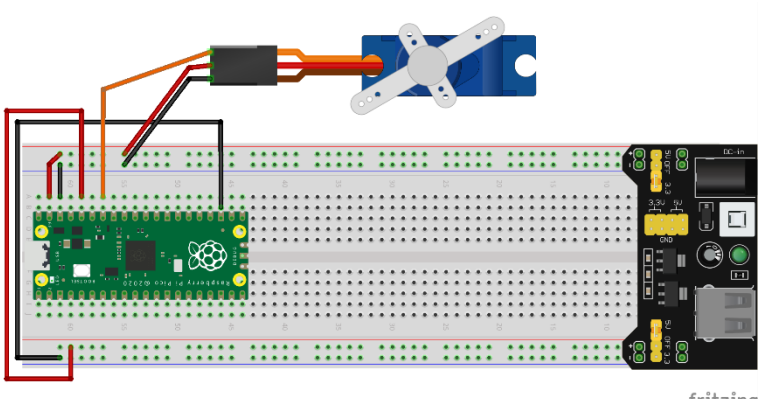
Descrição

Neste tutorial, você aprenderá como conectar e controlar um servomotor. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `servo.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x SG-90 servo motor

Diagrama de ligação



– cabo vermelho é conectado ao trilho de 5V (+)

– cabo preto / marrom está conectado ao trilho GND (-)

– cabo laranja é conectado ao pino GPIO28 ADC

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /servo.py @ 13 : 20

File Edit View Run Tools Help

servo.py

```
1  from machine import Pin, PWM #import Pin and PWM from library machine
2  from time import sleep  #import sleep from library time
3
4  pwm = PWM(Pin(28))
5  pwm.freq(50)
6
7  while True:
8      for position in range(1000,9000,50):
9          pwm.duty_u16(position)
10         sleep(0.01)
11     for position in range(9000,1000,-50):
12         pwm.duty_u16(position)
13         sleep(0.01)
```

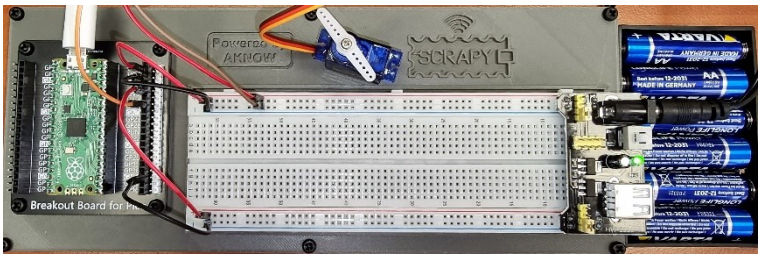
Shell

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

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



7. Ecrã OLED I2C SSD1306

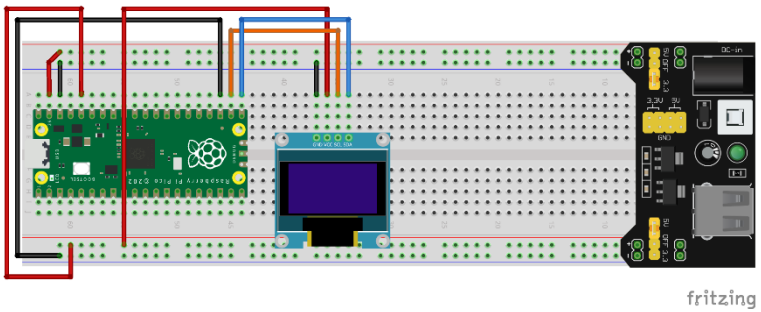
Descrição

Neste tutorial, você aprenderá como conectar e controlar o display I2C ICC OLED. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `oled.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 4 x Cabos jumper macho-macho
- 1 x Ecrã OLED I2C SSD1306

Diagrama de ligação



– cabo vermelho é conectado ao trilho 3v3 (+)

– cabo preto está conectado ao trilho GND (-)

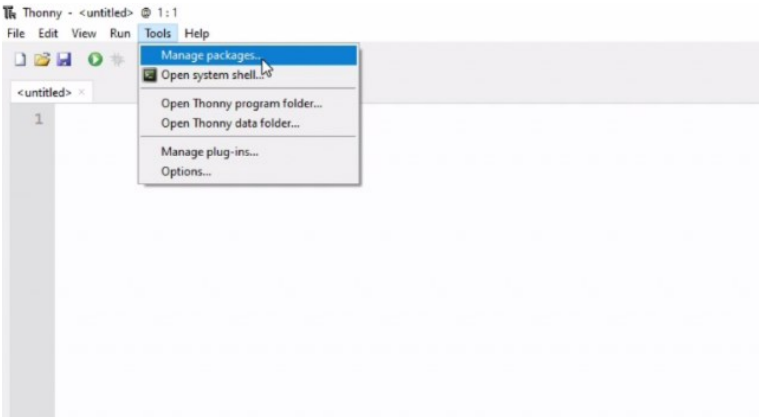
– cabo laranja é conectado ao pino GPIO17 I2C0 SCL

– cabo azul está conectado ao pino GPIO26 I2C0 SDA

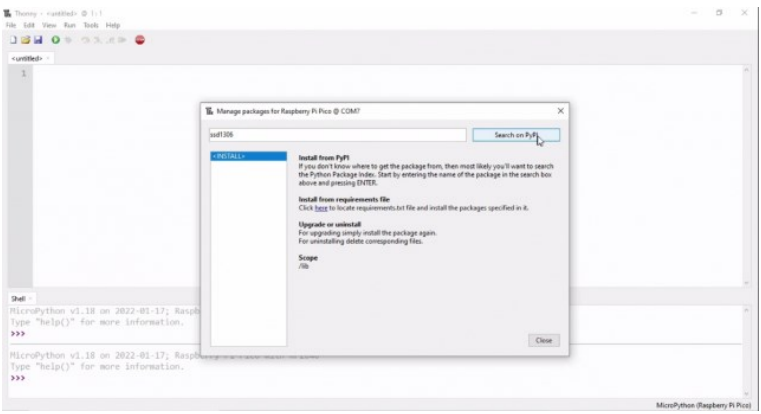
Código

Antes de iniciar a programação, o display OLED, primeiro precisamos adicionar o pacote SSD1306 ao nosso RPi Pico. Para fazer isso, siga as próximas etapas:

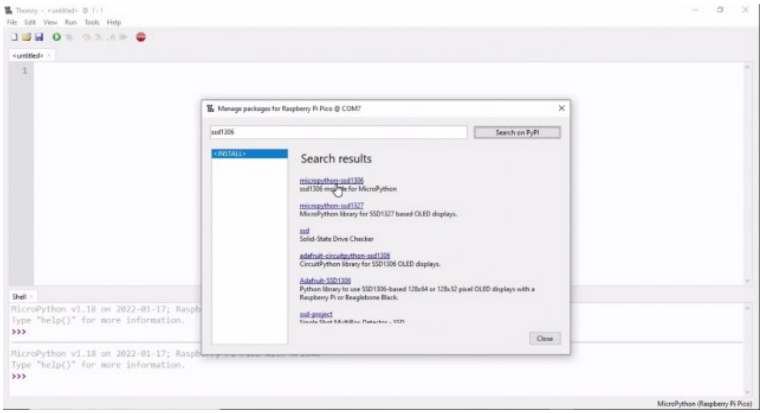
1. Abra o Thonny e vá para **Ferramentas → Gerenciar pacotes...**



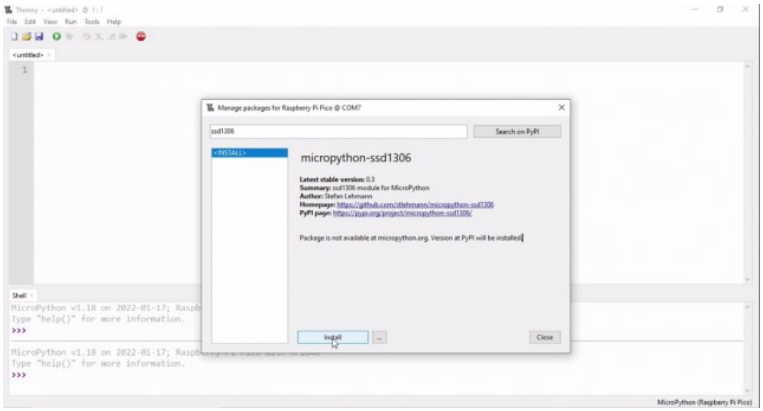
2. Na janela gerenciar pacotes, digite **SSD1306** e clique em *Pesquisar*.



3. Uma vez terminada a pesquisa, clique no *micropython-ssd1306.*



4. Na janela seguinte, clique em *Instalar.*



5. Espera pela instalação do pacote, clique em Fechar.

Agora, estamos prontos para prosseguir com a programação do display OLED.

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /oled.py @ 20 : 1

File Edit View Run Tools Help

[oled.py]

```
1  from machine import Pin, I2C
2  import ssd1306
3
4  WIDTH =128
5  HEIGHT = 64
6
7  PIN_SCL = 17
8  PIN_SDA = 16
9
10 i2c = I2C(0,scl=Pin(PIN_SCL),sda=Pin(PIN_SDA))
11 oled = ssd1306.SSD1306_I2C(WIDTH,HEIGHT,i2c)
12 oled.fill(0)
13
14 oled.text("OLED", 0, 0)
15 oled.text("Display", 0, 10)
16 oled.text("Hello SCRAPY", 0, 30)
17 oled.text("  People!", 0, 40)
18
19 oled.show()
20
```

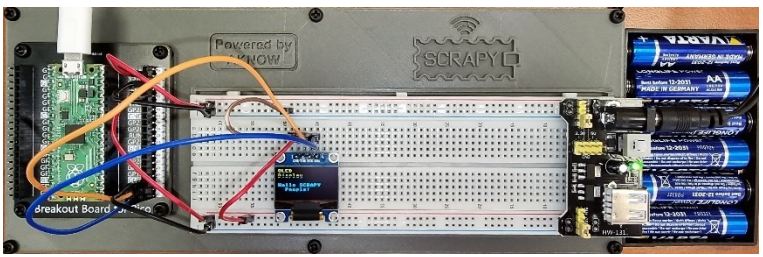
Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
>>>
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



8. Módulo joystick

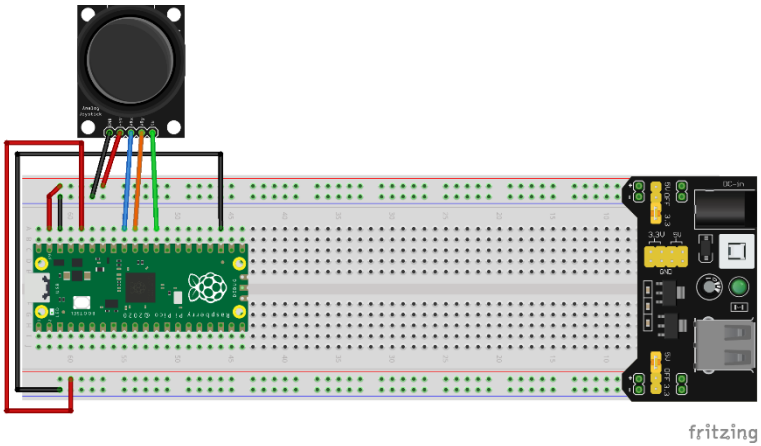
Descrição

Neste tutorial, você aprenderá como conectar e controlar um módulo de joystick. Abra Thonny Python, em seguida, vá para File → Save as..., escolha Raspberry Pi Pico, e salve seu arquivo sob o nome `joystick.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completa
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 5 x Cabos jumper macho-macho
- 1 x Módulo joystick

Diagrama de ligação



– +5V (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– VRx (cabo azul) está conectado ao pino GPIO27 ADC1

– VRy (cabo laranja) está conectado ao pino GPIO26 ADC0

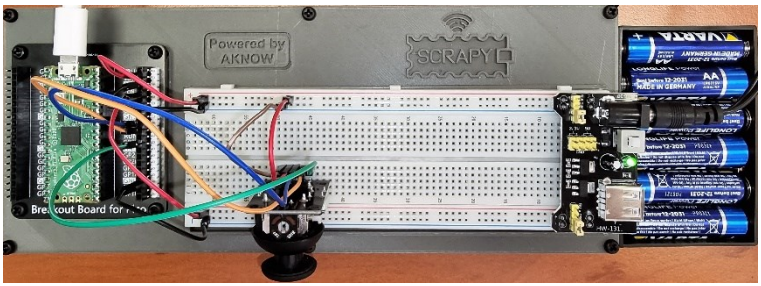
– SW (verde) cabo está conectado ao pino GPIO22

Código

Código MicroPython para o tutorial:

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



Tutoriais com sensores

9. Sensor de gota de chuva

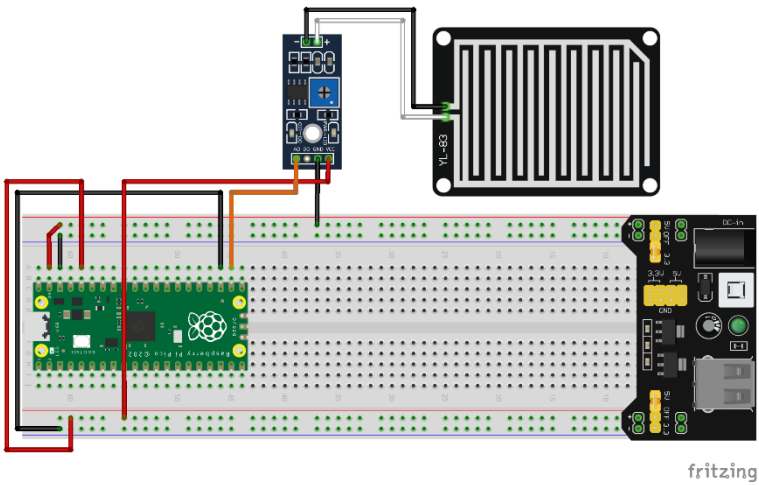
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de gota de chuva. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `raindrop.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard compelto
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x Sensor de gota de chuva

Diagrama de ligação



– 3v3V (cabo vermelho) está conectado ao trilho 3v3V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– DO (cabo laranja) está conectado ao pino GPIO17

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /raindrop.py @ 15 : 1

File Edit View Run Tools Help

[raindrop.py]

```
from machine import Pin
from time import sleep

PIN_RAIN = 17

rainSensor = Pin(PIN_RAIN, Pin.In)

while True:
    if rainSensor.value() == 1:
        print("Not raining")
        sleep(0.5)
    if rainSensor.value() == 0:
        print("It's raining!")
        sleep(0.5)

```

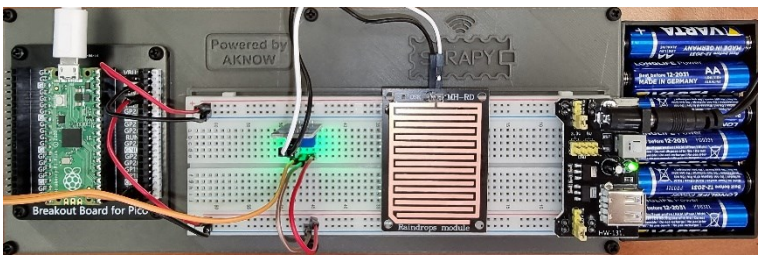
Shell

```
MicroPython v1.20.0 on 2023-04-26; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



10. HC-SR04 Sensor ultra-sónico

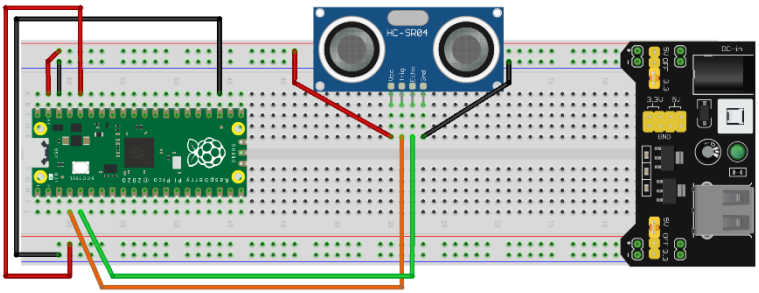
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor ultrassônico HC-SR04. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `ultrasonic.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1xHC-SR04 Sensor ultra-sónico

Diagrama de ligação



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

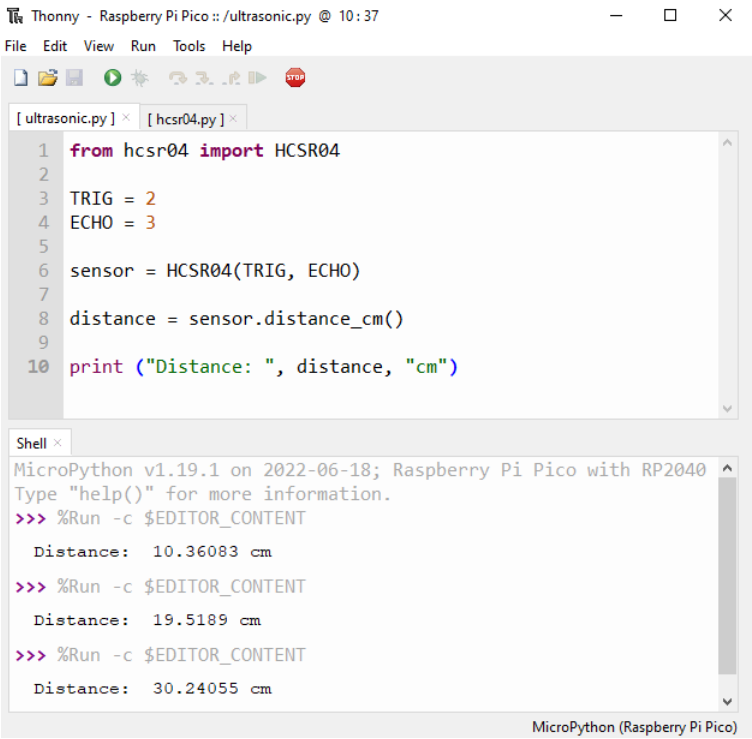
– TRIG (cabo laranja) está conectado ao pino GPIO2

– ECHO (cabo verde) está ligado ao pino GPIO3

Código

Para usar o sensor ultra-sônico HC-SR04, podemos desenvolver nosso próprio programa, ou usar uma das bibliotecas disponíveis on-line, por exemplo, no Github. Se optar por descarregar a biblioteca *hcsr04.py*, deverá guardá-la no seu Pico com esse mesmo nome.

Código MicroPython usando uma biblioteca existente:



Código MicroPython sem biblioteca existente:

Thonny - Raspberry Pi Pico :: /ultrasonic.py @ 21 : 18

File Edit View Run Tools Help

[ultrasonic.py]

```
from machine import Pin
import utime

TRIG = Pin(2, Pin.OUT)
ECHO = Pin(3, Pin.IN)
def ultra():
    TRIG.low()
    utime.sleep_us(2)
    TRIG.high()
    utime.sleep_us(5)
    TRIG.low()
    while ECHO.value() == 0:
        signaloff = utime.ticks_us()
    while ECHO.value() == 1:
        signalon = utime.ticks_us()
    timepassed = signalon - signaloff
    distance = (timepassed * 0.0343) / 2
    print("The distance from object is ",distance,"cm")
while True:
    ultra()
    utime.sleep(1)
```

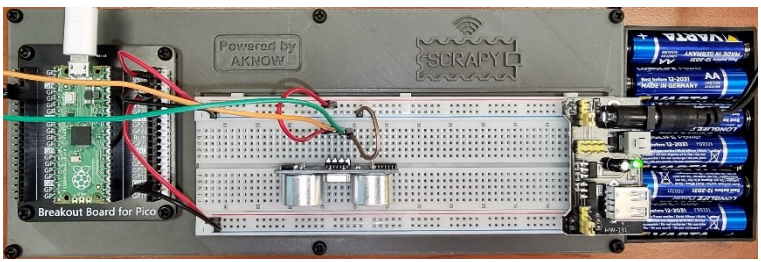
Shell

```
The distance from object is  153.9212 cm
The distance from object is  12.91395 cm
The distance from object is  22.8095 cm
The distance from object is  155.0874 cm
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



11. Sensor de movimento PIR

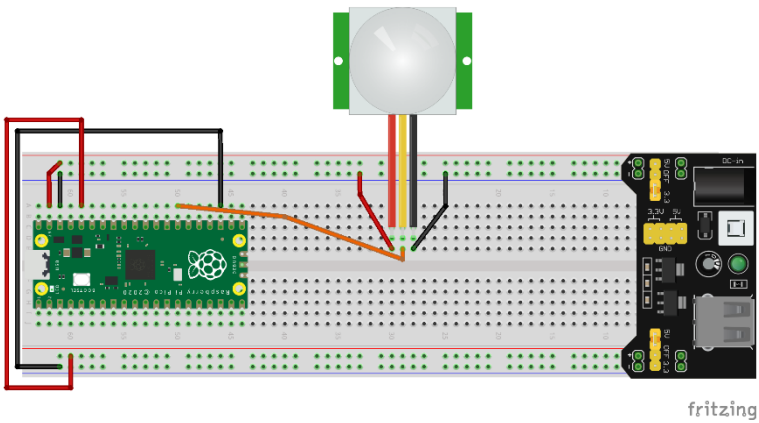
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de movimento PIR. Abra o Thonny Python e, em seguida, vá para Salvar arquivo → como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `motion.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-fêmea
- 1 x Sensor de movimento PIR

Wiring diagram



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– OUT (cabo laranja) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /motion.py @ 6 : 12

File Edit View Run Tools Help

[motion.py]

```
1  from machine import Pin
2  from time import sleep
3
4  PIR = Pin(21, Pin.IN, Pin.PULL_UP)
5
6  while True:
7      if PIR.value() == 1:
8          print("Motion detected!")
9          sleep(1)
10     if PIR.value() == 0:
11         print("No motion detected!")
12         sleep(1)
```

Shell

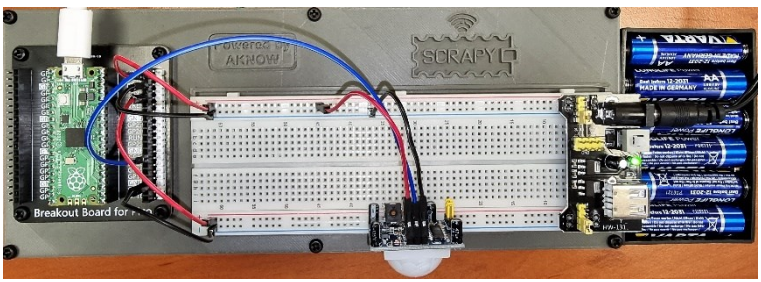
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  No motion detected!
  No motion detected!
  Motion detected!
  Motion detected!
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



12. Sensor DHT11

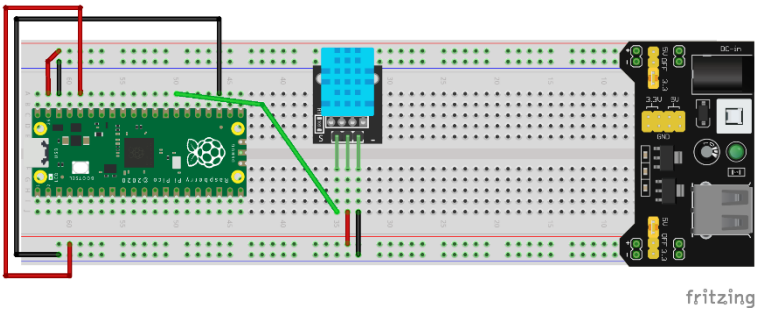
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de temperatura e umidade DHT11. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `dht11.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit Breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x Sensor temperatura DHT11

Diagrama de ligação



– VCC (cabo vermelho) está conectado ao trilho 3v3 (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– S (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /dht11.py @ 12 : 38

File Edit View Run Tools Help

[dht11.py]

```
1  from machine import Pin
2  from time import sleep
3  import dht
4
5  sensor = dht.DHT11(Pin(21))
6
7  while True:
8      sensor.measure()
9      temp = sensor.temperature()
10     hum = sensor.humidity()
11     print("Room temperature: ",temp, "Celcius")
12     print("Room humidity: ",hum, "%")
13     sleep(5)
```

Shell

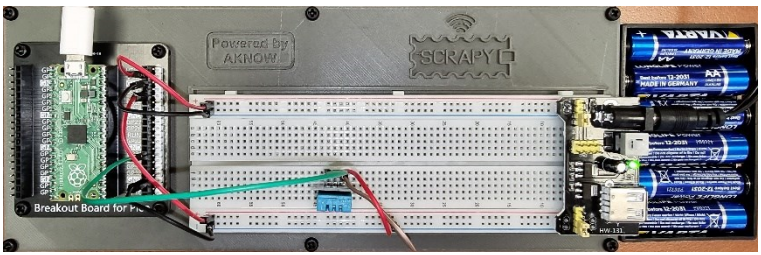
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Room temperature:  27 Celcius
  Room humidity:  34 %
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



13. Sensor de vibração SW-420

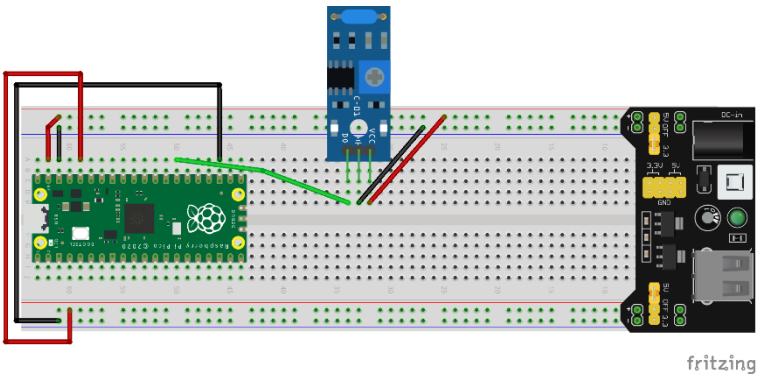
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de vibração SW-420. Abra Thonny Python e, em seguida, vá para File → Save as..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `vibration.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x SW-420 sensor de vibração

Wiring diagram



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– DO (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /vibration.py @ 8 : 37

File Edit View Run Tools Help

[vibration.py]

```
1  from machine import Pin
2  from time import sleep
3
4  vibration = Pin(21, Pin.IN, Pin.PULL_UP)
5
6  while True:
7      if vibration.value() == 0:
8          print("Vibration detected!")
```

Shell

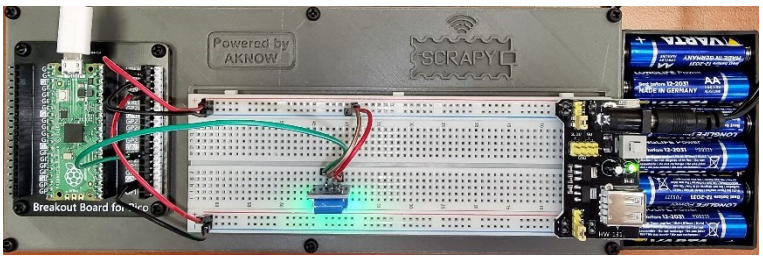
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Vibration detected!
  Vibration detected!
  Vibration detected!
  Vibration detected!
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



14. Sensor de chama

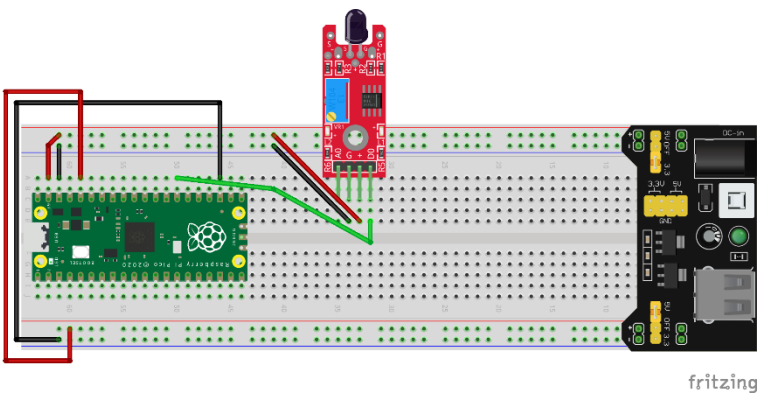
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de chama KY-026. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `flame.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x Sensor de chama KY-026

Wiring diagram



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– DO (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /flame.py @ 10 : 16

File Edit View Run Tools Help

[flame.py]

```
1  from machine import Pin
2  from time import sleep
3
4  flame = Pin(21, Pin.IN)
5  sleep(2)
6
7  while True:
8     if flame.value() == 0:
9          print("Flame Detected")
10         sleep(3)
```

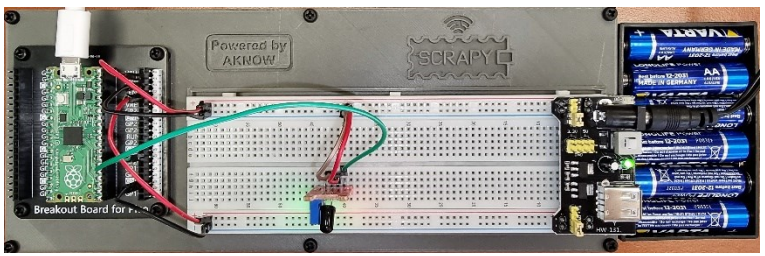
Shell

```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



15. Sensor de deteção de som

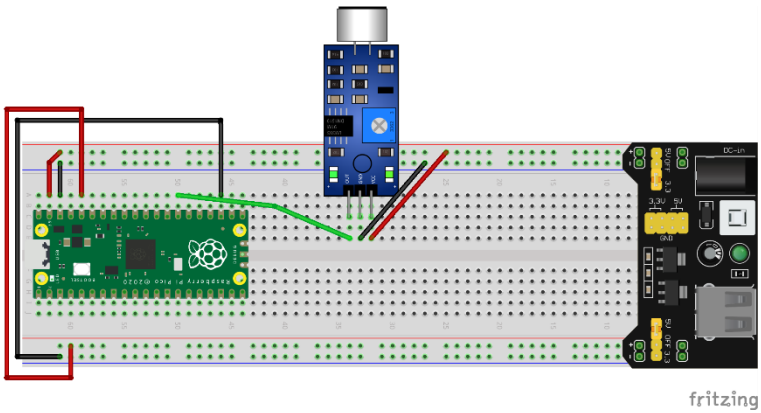
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de deteção de som KY-037. Abra Thonny Python e, em seguida, vá para Arquivo → Salvar como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `sound.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x Sensor de som KY-037

Diagrama de ligação



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– DO/OUT (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /sound.py @ 10 : 16

File Edit View Run Tools Help

[sound.py]

```
1  from machine import Pin
2  from time import sleep
3
4  sound = Pin(21, Pin.IN)
5  sleep(2)
6
7  while True:
8     if sound.value() == 0:
9          print("Sound Detected")
10         sleep(3)
```

Shell

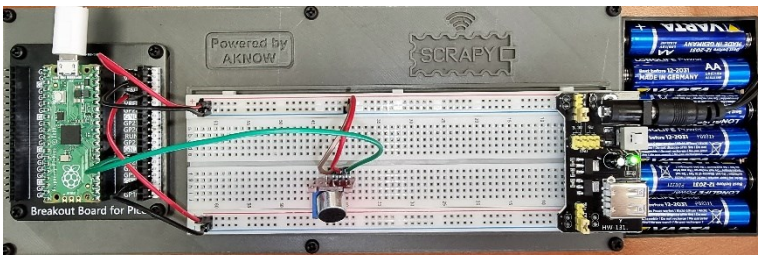
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Sound Detected
  Sound Detected
  Sound Detected
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



16. Sensor de humidade do solo

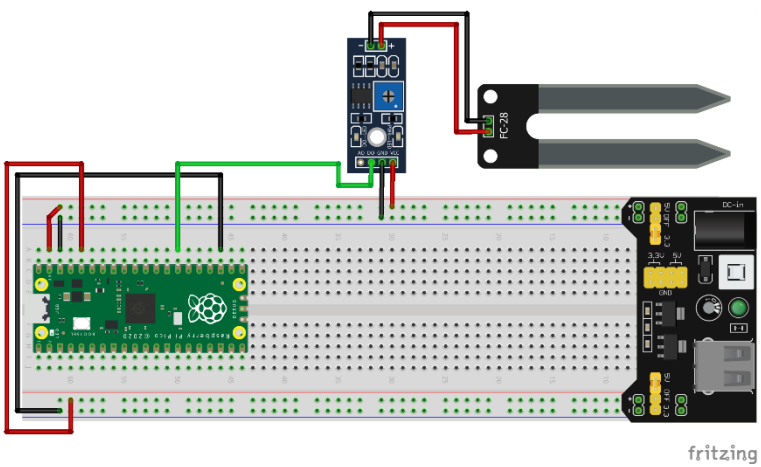
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor de umidade do solo. Abra Thonny Python, em seguida, vá para File → Save as..., escolha Raspberry Pi Pico, e salve seu arquivo sob o nome `soil.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo Micro-USB
- 1 x Kit breadboard Pico
- 3 x Cabos jumper macho-macho
- 1 x Sensor de som KY-037

Wiring diagram



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– DO/OUT (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /soil.py @ 10 : 16

File Edit View Run Tools Help

[soil.py]

```
1  from machine import Pin
2  from time import sleep
3
4  soil = Pin(21, Pin.IN)
5  sleep(2)
6
7  while True:
8     if soil.value() == 0:
9         print("Moisture Detected")
10        sleep(3)
```

Shell

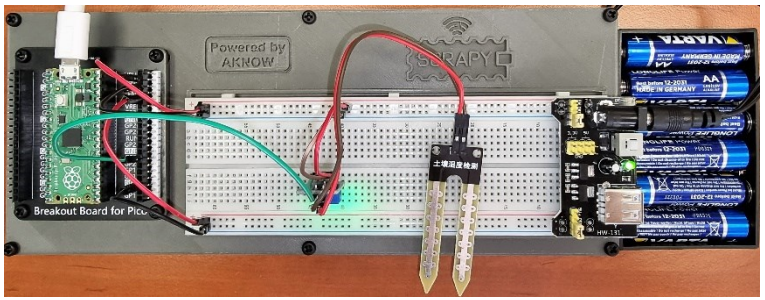
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Moisture Detected
  Moisture Detected
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



17. Sensor IR infravermelho

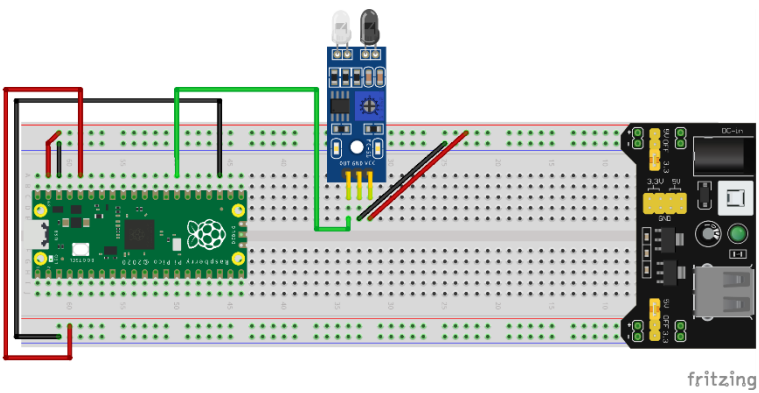
Descrição

Neste tutorial, você aprenderá como conectar e controlar o sensor infravermelho IR. Abra Thonny Python e, em seguida, vá para Salvar → arquivo como..., escolha Raspberry Pi Pico e salve seu arquivo sob o nome `ir.py`. Então é hora de conectar a eletrônica e escrever o seu programa. Por favor, siga as instruções abaixo.

Material necessário

- 1 x Raspberry Pi Pico
- 1 x Breadboard completo
- 1 x Cabo micro-USB
- 1 x Kit breadboard Pico
- 3 x cabos jumper macho-macho
- 1 x Sensor IR infravermelho

Wiring diagram



– VCC (cabo vermelho) está conectado ao trilho de 5V (+)

– GND (cabo preto) está ligado ao trilho GND (-)

– OUT (cabo verde) está conectado ao pino GPIO21

Código

Código MicroPython para o tutorial:

Thonny - Raspberry Pi Pico :: /ir.py @ 13 : 17

File Edit View Run Tools Help

[ir.py]

```
1  from machine import Pin
2  from time import sleep
3
4  ir = Pin(21, Pin.IN)
5  sleep(2)
6
7  while True:
8     if ir.value() == 1:
9          print("No obstacles")
10         sleep(3)
11    else:
12         print("Obstacle detected!")
13         sleep(3)
14
```

Shell

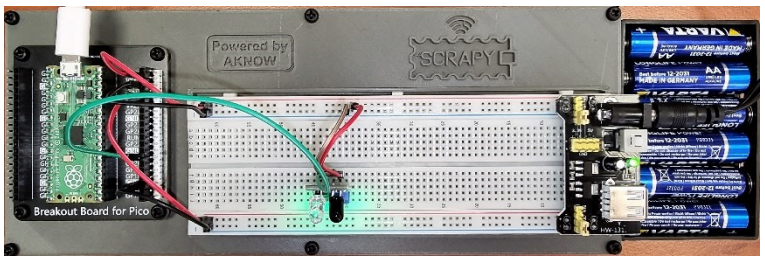
```
MicroPython v1.19.1 on 2022-06-18; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  Obstacle detected!
  Obstacle detected!
```

MicroPython (Raspberry Pi Pico)

Imagem de exemplo

Imagem de como o tutorial parece usando o hardware fornecido:



Apendice: Tabela de resumo do MicroPython

Output Digital		
Chamando a classe Pin	`from machine import Pin`	
Inicialização do objeto de saída digital	`led = Pin(pin_value, Pin.OUT)`	`pin_value` entre 0 e 40
Saída digital aberta (saída de 3,3 V)	`led.value(1)`	ON
Fechar saída digital (saída 0V)	`led.value(0)`	OFF

Input Digital		
Chamando a classe Pin	`from machine import Pin`	
Inicialização do objeto de saída digital	`button = Pin(pin_value, Pin.IN)`	`pin_value` entre 0 e 40
	`button = Pin(pin_value), Pin.IN, Pin.PULL_UP)`	Ativação da resistência PULL UP
	`button = Pin(pin_value), Pin.IN, Pin.PULL_DOWN)`	Ativação da resistência PULL DOWN
Leitura de entrada	`value = button.value(1)`	O valor de retorno pode ser 0 se o pino estiver em 0V, ou 1 se o pino estiver em 3,3V

Output Analógico (Modulação da largura de pulsação - PWM)		
Chamando a classe PWM	`from machine import PWM`	
Inicialização da saída analógica	`led = PWM(Pin(pin_value), frequency)`	`pin_value` entre 0 e 40 `frequency` em HZ, de 0 a 78125
Leitura de entrada	`led.duty(duty_cycle)`	`duty_cycle` de 0 a 1023 (saída de 0V a saída de 3,3V, respectivamente)

Input Analógico		
Chamando a classe ADC	`from machine import ADC`	
Inicialização da entrada analógica	`pot = ADC(Pin(pin_value))`	`pin_value` pode ser GPIO26, GPIO27 e GPIO28
Declaração em que tensão a entrada dará o seu valor máximo (em ESP32 geralmente 3.3V)	`pot.atten(ADC.ATTN_11DB)`	**ADC.ATTN_0DB:** tensão de gama completa: 1,2 V **ADC.ATTN_2_5DB:** tensão de gama completa: 1,5 V **ADC.ATTN_6DB**: tensão de gama completa: 2,0 V **ADC.ATTN_11DB**: tensão de gama completa: 3,3 V
Declaração do intervalo de valores de entrada (padrão de 12 bits)	`pot.width(ADC.WIDTH_10BIT)`	**ADC.WIDTH_9BIT**: intervalo de 0 a 511 **ADC.WIDTH_10BIT**: intervalo de 0 a 1023 **ADC.WIDTH_11BIT**: intervalo de 0 a 2047 **ADC.WIDTH_12BIT**: intervalo de 0 a 4095
Leitura de entrada	`value = pot.readl()`	`value` é um número inteiro de 0 até ao máximo do intervalo especificado pelo ADC. WIDTH_#BIT (ver anterior)

A biblioteca do tempo		
Chamando a classe do sono	`from time import sleep`	
Utilização da função do sono	`sleep(sec)`	`seg` é o número de segundos para os quais o programa será atrasado
Chamando a classe de tempo	`import time`	
Utilização da função de tempo	`current_time = time()`	A variável `current_time` terá um valor numérico, igual ao número de segundos desde a última reposição na placa.

Estrutura da instrução If	

`if <expr1>:`
`  <statement1>`
`elif <expr2>:`
`  <statement2>`
`elif <expr3>:`
`  <statement3>`
`  (...)`
`else:`
`  <statementn>`
	<expr#>: a condição de controle que deve retornar True ou False
	
	<statement#>: conjunto de comandos a serem executados quando a condição adjacente for satisfeita
	
	<expr#> (e.g. the set <statement2> é executado quando <expr2> é satisfeito)
	
	<statementn>: Conjunto de instruções executadas quando nenhuma das condições <expr#> é satisfeita

Estrutura da instrução While em loop	
`while <expr>:` `  <statement(s)>`	<expr>: a condição de controle que deve retornar True ou False <statement#>: conjunto de comandos a serem executados desde que a condição <expr> seja satisfeita

Para estrutura de loop	
`for <var> in <iterable>:` `  <statement(s)>`	**<iterable>**: uma coleção de objetos, por exemplo, uma lista contendo números, alfanuméricos, etc. **<var>**: a variable to which the value of the next item in the collection <iterable> is assigned. **<stametement(s)>**: um conjunto de instruções executadas em cada iteração
`for <var> in range(<start>, <end>, <step>):` `  <statement(s)>`	**range(<start>, <end>, <step>)**: função que retorna uma sequência de números de <start> para <end>-1, com uma diferença de <step> entre dois números consecutivos (<start> e <step> parâmetros são opcionais). **<var>**: variável à qual é atribuído o valor do próximo elemento da sequência produzida pelo intervalo.

		<statement(s)>: um conjunto de instruções executadas em cada iteração
Miscellaneous		
DHT11	`import dht`	Importando a biblioteca DHT
	`sensor = dht.DHT11(Pin(pin_number))`	Inicialização da variável sensor com sua pin_number associada .
	`sensor.measure()`	Atualização dos valores do sensor
	`temp = sensor.temperature()`	Poupança do valor da temperatura atual
	`hum = sensor.humidity()`	Poupança do valor da humidade atual
ECRÃ OLED	`from machine import I2C`	Importando a biblioteca I2C
	`import ssd1306`	Importando a biblioteca ssd1306
	`i2c = I2C(-1, scl=Pin(1), sda=Pin(0))`	Inicialização da variável i2c nos pinos SCL & SDA do Pico

	`oled_width = 128`
	`oled_height = 64`
	`oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)`
		Inicializando a tela

	`oled.text('Hello, World 1!', 0, 0)`
	`oled.text('Hello, World 2!', 0, 10)`
	`oled.text('Hello, World 3!', 0, 20)`
		Armazenando mensagens no buffer de tela

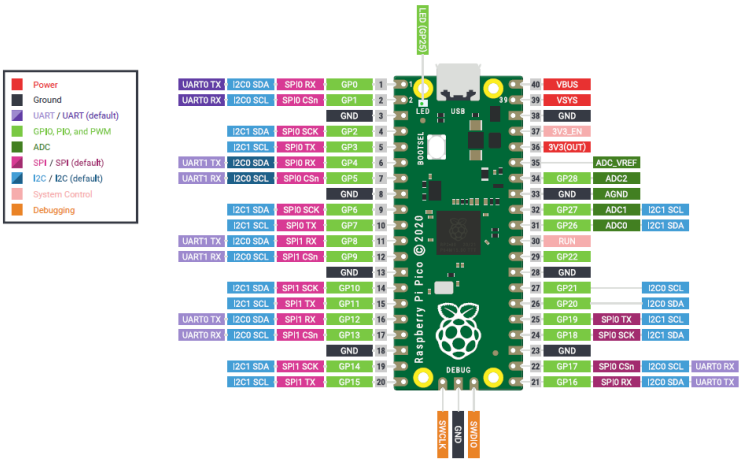
	`oled.show()`	Mostrando mensagens (necessário para exibir mensagens armazenadas no buffer de tela)
	`display.pixel(3, 4, 1)`	Defina o pixel localizado na posição (x,y) na tela, com x=3 & y=4, para o estado 1 (ou seja, exibição)

Co-funded by
the European Union

SCRAPY

2021-1-FR01-KA220-SCH-000031617

Raspberry Pi Pico Pinout



Raspberry Pi

w: scrapykit.eu
e: info@scrapykit.eu